CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Unitn-eprints PhD

UNIVERSITY OF TRENTO

DEPARTMENT OF MATHEMATICS

Doctoral School in Mathematics

PhD Thesis

Formal Proofs of Security for Privacy-Preserving Blockchains and other Cryptographic Protocols

Advisor:
Prof. Massimiliano Sala

Graduand:
Riccardo Longo

First Reader:
Prof. Massimo Bartoletti

Second Reader
Prof. Andrea Bracciali

Academic Year 2016 - 2017

Formal Proofs of Security for Privacy-Preserving Blockchains and other Cryptographic Protocols

Riccardo Longo

March 9th 2018

Contents

Abstract

Cryptography is used to protect data and communications. The basic tools are cryptographic primitives, whose security and efficiency are widely studied. But in real-life applications these primitives are not used individually, but combined inside complex protocols. The aim of this thesis is to analyse various cryptographic protocols and assess their security in a formal way.

In chapter 1 the concept of *formal proofs of security* is introduced and the main categorisation of attack scenarios and types of adversary are presented, and the protocols analysed in the thesis are briefly introduced with some motivation.

In chapter 2 are presented the security assumptions used in the proofs of the following chapters, distinguishing between the hardness of algebraic problems and the strength of cryptographic primitives.

Once that the bases are given, the first protocols are analysed in chapter 3, where two *Attribute Based Encryption* schemes are proven secure. First context and motivation are introduced, presenting settings of cloud encryption, alongside the tools used to build ABE schemes. Then the first scheme, that introduces multiple authorities in order to improve privacy, is explained in detail and proven secure. Finally the second scheme is presented as a variation of the first one, with the aim of improving the efficiency performing a round of collaboration between the authorities.

The next protocol analysed is a *tokenization algorithm* for the protection of credit cards. In chapter 4 the advantages of tokenization and the regulations required by the banking industry are presented, and a practical algorithm is proposed, and proven secure and compliant with the standard.

In chapter 5 the focus is on the *BIX Protocol*, that builds a chain of certificates in order to decentralize the role of certificate authorities. First the protocol and the structure of the certificates are introduced, then two attack scenarios are presented and the protocol is proven secure in these settings. Finally a viable attack vector is analysed, and a mitigation approach is discussed.

In chapter 6 is presented an original approach on building a public ledger with *end-to-end encryption* and a *one-time-access* property, that make it suitable to store sensitive data. Its security is studied in a variety of attack scenarios, giving proofs based on standard algebraic assumptions.

The last protocol presented in chapter 7 uses a *proof-of-stake* system to maintain the consistency of subchains built on top of the Bitcoin blockchain, using only standard Bitcoin transactions. Particular emphasis is given to the analysis of the refund policies employed, proving that the naive approach is always ineffective whereas the chosen policy discourages attackers whose stake falls below a threshold, that may be adjusted varying the protocol parameters.

Finally some conclusions are drawn in chapter 8.

Chapter 1

Introduction to Formal Proofs of Security

Cryptographic protocols protect the exchanges of information that permeate our lives. In the modern world countless scenarios need a secure way to communicate, each with its own requirements, giving birth to many new protocols. The security of these protocols relies not only on the soundness of their building blocks, but also on how all the primitives work together and the parameters are chosen. To have a reasonable confidence in their security, these protocols are analysed from a formal point of view, where carefully modelled attacks are compared to well studied mathematical problems.

1.1 Security of Cryptographic Protocols

Information has been a valuable commodity since the dawn of history, but the digital revolution has increased its role and importance to the point that a great deal of our everyday activities revolves around information exchanges. Given its primary role, the protection of such data interchange is of paramount importance. Moreover the same technology that allows an easy distribution of data, promotes wide diffusion of knowledge, and shortens the distance between people also facilitates the collection of the same data, consents eavesdropping on unprecedented scales and opens the door to a variety of intrusions and violations of personal privacy.

To our rescue comes cryptography, the art of hiding messages from privy eyes. Modern ciphers achieve great efficiency and effectiveness in protecting the content of our communication, but in real life applications privacy is not the only factor: in many scenarios authenticity and integrity are more relevant than secrecy. For this purpose new primitives have been developed and then combined into complex protocols. Thanks to these protocols we now can safely connect our phone to a server on the other side of the world and purchase anything from books to furniture, pay our taxes from the comfort of our homes, receive *over the air* updates for our new "smart car". It is obvious that all these operations involve sensible exchanges of information: we do not want our credit card number stolen, we do not want a identity thief to steal our personal data, and we surely do not want malicious software to find its way into our car

compromising its functionality and security.

1.2 Simulator and Adversary: the General Structure of a Proof

In cryptography the security of a scheme usually relies on the hardness of a particular mathematical problem. So, in a formal proof of security, the goal is to model the possible attacks on the scheme and prove that a successful breach implies the solution of a hard, well-known mathematical problem. Some security parameters may be chosen in such a way that the problem that guarantees the security becomes almost impossible to solve (in a reasonable time), and thus the scheme becomes impenetrable.

More formally, the scheme is supposed secure if an *Assumption* holds on the related mathematical problem. Generally an *Assumption* is that there is no *polynomial-time* algorithm that solves a problem $\mathcal{P}$ with *non-negligible* probability. See Chapter 2 for some examples.

In a formal proof of security of a cryptographic scheme there are two parties involved: a *Challenger* $\mathcal{C}$ that runs the algorithms of the protocol and an *Adversary* $\mathcal{A}$ that tries to break the scheme making queries to $\mathcal{C}$. In a *query* to $\mathcal{C}$, depending on the security model, $\mathcal{A}$ may request private keys, the encryption of specific plaintexts, the decryption of some ciphertexts and so on. The goal of $\mathcal{A}$ also depends on the security model, for example it may be to recover a key, to forge a digital signature, to invert a hash function, or another similar purpose.

Hence, security proofs follow the following general path. Suppose there is an Adversary $\mathcal{A}$ that breaks the scheme with non-negligible probability p_1. A *Simulator* $\mathcal{S}$ is built in such a way that if $\mathcal{A}$ breaks the scheme then $\mathcal{S}$ solves $\mathcal{P}$. So, given an instance of $\mathcal{P}$, $\mathcal{S}$ runs a challenger $\mathcal{C}$ that interacts with $\mathcal{A}$, simulating the scheme correctly with non-negligible probability p_2. Thus $\mathcal{S}$ solves $\mathcal{P}$ with non-negligible probability (usually $p_1 p_2$), contradicting the Assumption.

To summarize, a formal proof of security is a *reduction* from the problem *attack the scheme* to the problem *solve* $\mathcal{P}$. Typically $\mathcal{P}$ is a well-studied problem so the assumption on its insolvability is accepted by the academic community.

1.3 Attack Scenarios and Security Properties

Evaluation and comparison between protocols from a functionality point of view is often difficult, because the different purposes or the efficiency trade-offs. From a security point of view instead we can evaluate protocols better in terms of the assumptions they rely on and the attack scenarios they can endure.

In this section we focus on the main types of attack scenarios, giving an idea of the level of security they refer to. In these scenarios an adversary interacts with the protocol trying to circumvent the defences and gain information that should not be revealed. Ideally a protocol should withstand a very powerful foe without revealing anything useful. So the two primary aspects that characterize a scenario are the strength of the adversary and its goal (i.e. which informations it tries to obtain).

1.3.1 Adversary Strength

When comparing the strength of the adversary in a security game, stronger foe is preferable, since a scheme that resists a mighty opponent can fend off attackers with less resources too, and obviously security is higher when fewer adversaries can effectively attack.

The strength of an adversary is determined by its computational power and the information it has access to (obtained either beforehand or interacting with the protocol). In many scenarios however the adversary is given unlimited computational power, so the distinction is based purely on the queries the attacker can send to the protocol and the data at its disposal. Clearly more queries mean more data to work on, and potentially more information leaked. So a good protocol should keep private content safe even when a lot of side data has been disclosed.

Ciphertext Only

In the weakest scenario the adversary has access to ciphertexts only, and tries to deduce from them what it needs to reach its goal (see Section 1.3.2).

This case models the most common situation in cryptography, where the communication channel is insecure thus an outsider can observe every message transmitted. In this case the interaction with the protocol is minimum. A possible further classification may count the number of ciphertexts the adversary can observe, modelling the case in which the encryption key changes after a certain amount of messages have been encrypted.

Known Plaintext

The next step sees an adversary that has access to some extra information. In this scenario the attack is performed with the additional knowledge of plaintext-ciphertext pairs. As in the previous case the number of these pairs at disposal may be considered as a sub-index of the strength of the adversary, where stronger ones have access to more pairs.

The situation modelled here is one where there are a number of known messages that will be encrypted and transmitted, or some data has been leaked. For example during World War II it has been essential in breaking enigma machine's encryption the role played by the so called *cribs*, that relied on known plaintexts such as `ANX` (*an* is German for *to*), `EINS` (*one*) and weather forecasts. These words were widely present across various communications, and this helped finding the encryption keys. Moreover various messages were sent using different ciphers, so once the weaker encryptions were decrypted the cryptanalysts had the plaintexts to help them find the keys.

Note that in this case the goal of the adversary cannot be the decryption of any ciphertext, because the availability of plaintext-ciphertext pairs could make this challenge trivial. So either the target is a ciphertext for which the decryption is not already known, or something more general, e.g. to find the encryption key.

Chosen Plaintext

In this case the adversary does not have more information in terms of quantity, but does have more valuable information because this time can choose some messages to be encrypted. The gain arises when there are some plaintexts that are easier to work on or that generate weaker encryptions that expose the key partially or totally.

Besides the usual discrimination in the number of queries allowed, there is a further distinction: the possibility of adaptive choice. Adaptive means that the attacker can chose some messages, see their encryption, choose some more and so on. In this case the adversary can look for increasingly weaker plaintext potentially leading to hindsights that can break the security. This procedure is in contrast to a weaker adversary that has to choose the messages before having access to any encryption.

To compare this model with a real-life scenario suppose that a spy has brief access to an encryption machine so that a few ciphertexts can be produced choosing the messages, or again suppose that some information is fed to the enemy so that it is transmitted encrypted to the high quarters. Both these cases have happened during World War II: a German spy was turned by the British intelligence with the main purpose of having her wordy reports encrypted with enigma machines, and a common practice to attack the naval codes was to plant mines in specific coordinates selected so that their encryptions would give the maximum help in recovering the keys.

Chosen Ciphertext

The strongest adversary has access to a very powerful tool: *decryption queries*. This means that this attacker can observe a number of ciphertexts and then request the decryption of some of them, obtaining the correspondent plaintext.

Again an adaptive adversary can make further queries after seeing some decryptions, instead of choosing the ciphertexts to decrypt beforehand. This means that smarter choices can be made, and the attack can be more effective with fewer queries.

A real life situation that may correspond to this model could see a spy having limited access to a decrypting machine to decipher some messages, or there is access to some messages after they have been decrypted thanks to a bug or a wiretap on an unencrypted internal line.

Note that in the case of *asymmetric ciphers* generally the adversary knows the public key, therefore has access to free unlimited encryption queries (since everyone that has the public key can compute the encryption by themself). This means that the known plaintext and chosen plaintext settings have little sense, therefore the need for a stronger adversary. However when the encryption is randomized (like in many asymmetric schemes) weaker attackers still make sense because the same plaintext can be encrypted into several different ciphertext. So a free *encrypt* query is not useful to check if a given message is the decryption of a certain ciphertext.

1.3.2 Adversary Goal

Concerning the goals of the adversary, the highest security comes when the goal of the adversary is the easiest. The idea is that when even the simplest attacks

can not be carried out then the protocol protects also against bigger threats.

In the hierarchy of the adversaries the stronger ones can do (and know) everything the weaker ones can, likewise a hierarchy of goals will be presented in the form of problems with decreasing difficulties. That is if an attacker can solve the problem that gives the weakest security, then it can solve every other problem. Vice versa an adversary that can not solve the problem related to the highest security level can not solve the other ones, and if someone can solve it is not necessary able to solve the lower ranking problems.

Basically the lowest problems are associated to the greatest disruption of the security: if they are solved the system is utterly compromised, whereas data could still be relatively safe even if a top-level problem can be solved.

Key Recovery

If an adversary is able to recover the master key of a protocol, then it is apparent that the system is completely compromised. With the key it is possible to encrypt and decrypt everything, thus the security is obliterated. So the basic requirement is that an attacker is not able to recover the key observing encrypted traffic, with optional access to other queries, as described in the previous section 1.3.1.

Note that some systems possess the so called *forward secrecy* property that guarantee the privacy of past communications when long-term keys are compromised. This is achieved using session keys that are discarded after use. Obviously this practice increases the security, but note that if an attacker can consistently recover session keys then this feature cannot salvage the security of the protocol.

This distinction is particularly important when the key can be compromised via other means, e.g. stealing the password from an unsafe storage or directly guessing it. In these cases forward secrecy can be useful because *the key is not leaked by the scheme itself*, so one can assume that a good password management can solve a breach, whereas if the protocol itself gives the key away to an attacker strong enough then no password policy can help.

Plaintext Recovery

The next step of security has what is probably the most common purpose for an adversary: decrypt the communication. This goal may be weakened by requiring the recovery of only a few specific plaintexts or a certain category of them.

Note that it is not always necessary to recover the key in order to decrypt the ciphertext. For example a stream cipher with short period uses the same keystream to encrypt different ciphertexts, so a known plaintext attack can effectively recover some messages without retrieving the actual key (the initialization vector). As a further remark on forward secrecy this kind of attacks effectively compromise the security despite the secrecy of the key, so it is important to clearly distinguish between the security of key and data.

Ciphertext Distinguisher

The top level aims to guarantee that no information about the plaintext is leaked once encrypted. In this case the adversary interacts with a challenger that is in

a state randomly chosen (there usually are two possible states), and the goal of the attacker is to distinguish between the two states, thus determining in which one the challenger is, hence the name. This state determines how the challenger runs the protocol and responds to the queries.

For example the property called *deniable encryption* requires that the ciphertext is indistinguishable from random noise. In this case the challenger may either be in the state *encrypt normally* or *output random noise*.

One of the most common distinguishing goals is to determine which of two equal-length messages has actually been encrypted. That is the attacker knows (or has directly chosen) two plaintexts, the challenger randomly selects one of these two messages and encrypts it. This ciphertext is given to the adversary that has to find out which plaintext is derived from.

Note that when the goal is to determine which state is correct among a finite set of possibilities, then the adversary can always win by guessing. Moreover in most cases the choice is limited between only two possibilities, so the attacker has at least fifty percent of chance to win. However this obviously does not mean that the protocol is insecure, but that the security estimate can not be measured in a single test. In fact the idea is to repeat the security game a sufficient number of times so that it is possible to estimate the winning probability of the adversary and confront it with a random guess. So the protocol is deemed insecure if it exists an attacker that has a sensible advantage in solving the challenge. That is, in the case where the adversary has to distinguish between two states, the probability of winning is:

$$p = \frac{1}{2} + \epsilon$$

where ϵ is non-negligible (see Definition 1.1) in function of a security parameter k. This parameter is used to scale the size of the various elements involved (keys, messages, ciphertexts, etc.) so that the security problem becomes increasingly difficult.

For example the security of *RSA protocol* is based on the problem of integer factorization, the security parameter k is the length of the key, and the difficulty is sub-exponential in this parameter: $\sim 2^{O(k^{\frac{1}{3}})}$.

So a protocol is secure against a distinguisher adversary if the advantage of the attacker is a negligible function in the security parameter.

Definition 1.1 (Negligible Function)**.** $\eta(k)$ is a negligible function in k if, for every c and for every γ there exists k_0 such that

$$|\eta(k)| < \left|\frac{1}{ck^{\gamma}}\right| \qquad \forall k > k_0. \tag{1.1}$$

1.4 The Protocols in this Thesis: a Motivation

This section gives a general review of the protocols presented and analysed in this thesis in the chapters 3 to 7, giving some motivation.

1.4.1 Attribute-Based Encryption

The first protocols, presented in chapter 3 fall in the category of *Attribute-Based Encryption*. These schemes address security in a context of shared resources.

That is files are used by multiple people and are therefore stored on a common space, but the content should be encrypted to protect data at rest, and not every user should be able to access every information, so keys should be carefully managed.

With the recent boom of cloud solutions, the potential of usage in real-life applications of these schemes is apparent, and justifies the great academic interest in this topic.

The original schemes presented in sections 3.3 and 3.4 analyse in particular a setting in which key management is distributed among multiple authorities, to avoid a single point of failure and enhance the effectiveness of security controls.

The first scheme (3.3) allows a variable number of independent authorities to be involved in the scheme, and gives the encryptor the possibility to choose the authorities to be considered trustworthy. Whereas the second scheme (3.4) fixes the set of authorities in advance and requires a collaboration round between them to combine their (independently chosen) parameters into a single set. These constraints however allow users to combine the individual keys into a single one and compress ciphertexts greatly, thus enhancing the efficiency significantly.

Both schemes are proved secure against an adversary that tries to distinguish between cipherte xts, having choice over the plaintexts (chosen-plaintext indistinguishability or IND-CPA). The proofs proceed to reduce an attack to scheme to the solution of an algebraic problem in the well-studied class of Diffie-Hellman problems.

1.4.2 Tokenization

In chapter 4 the proposed protocol addresses the security of credit card numbers. With the rise of e-commerce and a general shift towards digital currency, credit cards are used more than ever, but they are also more vulnerable due to the easiness of exploitation and careless behaviours (such as photos where PAN number is visible). To protect cardholders the tokenization process generates an alternative card number (called *token*) with limited validity to be used instead of the "proper" one.

The protocol explained in section 4.3 respects every requirement of the PCI-DSS industry standard and generates a token with the same format of a normal PAN, thus ensuring compatibility with every payment system.

The protocol is proved secure against chosen-plaintext indistinguishability attacks (IND-CPA). The proof (4.4) reduces an attack to the protocol that successfully recognises which of the chosen PANs has generated the output token, to an attack to the symmetric cipher used for the tokenisation, specifically AES.

1.4.3 BIX Certificates

The next protocol, presented in chapter 5 proposes an alternative Public Key Infrastructure (PKI) that decentralises the role of Certificate Authorities (CAs) using a blockchain-like approach. PKI has a fundamental role in securing internet navigation, but the current system based on CAs has significant limitations, as exemplified by the numerous attacks and breaches in the recent years.

In the BIX protocol outlined in section 5.1 identity certificates are cross-signed by users and stored in a blockchain-like data structure that ensures their

integrity and avoids a single point of failure.

The security analysis proves the impossibility of static tampering of the certificates if the underlying cryptographic primitives (hash function and digital signature) are secure. First an attack that extends the chain of certificates without proper interaction with the protocol is reduced to an attack to a crypto primitive (5.2). Then a similar technique is used for an attack that meddles with an internal certificate (5.3), combining the results it is clear that no external attacker can alter the certificates. However in section 5.4 it is showed how internal collusion could create alternative chain segments, and some mitigations are suggested.

1.4.4 Public Ledger for Sensitive Data

In chapter 6 an original protocol offers a solution for a public storage with verifiable integrity enhanced with strong privacy that makes it suitable for sensitive data.

Personal data should always be protected from privy eyes, but nonetheless it is widely used in a myriad of services and is necessarily shared with many companies. Moreover many applications require data integrity and verifiability (for example insurance companies despise tampering), that unfortunately is rather difficult to achieve while preserving privacy. The protocol prosed in section 6.2 builds a public ledger where integrity is publicly verifiable, but suitable for sensitive data storage, since the content is encrypted end-to-end, that is the owner may share its documents with any service provider, but no intermediary can see any content. Moreover access to data is periodically revoked enhancing privacy and control.

The security analysis considers various internal and external adversaries, and proves the security reducing the attacks to solution of Diffie-Hellman problems (6.4).

1.4.5 Proof of Stake Protocol for Bitcoin Subchains

The last protocol analysed in chapter 7 aims to preserve the consistency of subchains built on top of the Bitcoin blockchain using a "proof-of-stake" consensus.

Bitcoin has gained great popularity and recognition, and the volume of resources spent globally for its operation makes its blockchain very valuable. The protocol explained in section 7.4 runs a subchain (usable e.g. for smart contracts) exploiting Bitcoin blockchain, and incentivises the consistency of the updates using a proof of stake consensus to vote the next update and a refund policy to encourage honesty.

The refund policy is the central subject of the security analysis made in section 7.5. It is showed how naïve solutions do not incentivise honest behaviour, and is proved instead the effectiveness of a harsh policy that burns the refund lest it refund a dishonest actor.

1.5 Caveat

Formal proofs of security give a concrete measure of the strength of a protocol, showing that in a particular scenario it is impossible to perform a successful

attack.

However these proofs are no silver bullet, in fact their results lean on assumptions that may not hold or be disrupted in a near future (for example quantum computers are supposed to be able to efficiently solve many classical *hard problems*). Moreover real world attacks could escape the restrictions of the model in which the protocol was proven, rendering the proof almost useless.

For example recently the *KRACK attack* on the WPA2 protocol that protects wifi communications [83] exploited a bug in the protocol to effectively render the encryption useless, despite the protocol being proven secure. This was possible because the attack scenario in the model did not cover the actions used in the actual attack. That is, an implicit assumption was used but never stated, and this caused the production of millions of devices that implement a protocol now susceptible to attacks. Luckily a simple patch applied on the client device is sufficient to prevent the attack, but this case remains an excellent caveat that security is a very delicate matter.

Moreover even with carefully designed and proven protocols, problems may arise in implementations: the history of security bugs is full of *overflow* and *boundary* errors. And a new frontier of attacks on cryptography relies on side channels, that is metrics such as power consumption, heat or ultrasound frequencies are analysed to infer the computations performed by the encrypting machine and thus reverse-engineering the key used. To contrast these techniques, standard mathematical proofs are not helpful and various strategies are adopted to mask the signals that may give away the keys.

Finally, as the saying goes, the problem often lies *between the chair and the keyboard*, meaning that the weakest link in security is in most cases the end user. From bad security practices to social engineering vulnerabilities, there are multiple ways with which targeted attacks can bypass security protocols completely.

However despair is never the solution, and the development of new schemes may help to improve the security of world's communications and data.

Chapter 2

Hard Problems and Cryptographic Assumptions

The core principle of cryptography is to process plaintexts in such a way that ciphertexts become next to impossible to unscramble (without the proper key). In mathematics there are many functions that are fairly easy to compute but very difficult to invert, so they are natural candidates to become the foundation of cryptographic primitives.

As seen in the previous chapter, cryptographic protocols may be considered secure if it is almost impossible for an attacker to bypass its defences and gain access to information they are not entitled to know. In a formal proof of security the goals and resources of an attacker are modelled in a game, then the proof shows that attackers cannot win unless they are able to solve a difficult, well studied mathematical problem.

In this chapter we present various problems that are considered hard to solve. These problems will be the foundation of the security of the schemes examined in this thesis.

2.1 Bilinear Groups and Diffie-Hellman Problems

Bilinear pairings are a powerful tool used to build many asymmetric encryption schemes with interesting functionalities. Among these protocols we find many *Attribute-Based Encryption schemes*. This section covers background information necessary to understand the *KP-ABE* schemes presented in Chapter 3 and their security.

In particular, some mathematical notions about bilinear groups are given, alongside the cryptographic assumptions that will be used.

2.1.1 Bilinear Maps

Pairing-based cryptography exploits the properties of bilinear pairings to add new functionalities to encryption schemes difficult or impossible to achieve with the classical primitives. Bilinear groups are the main environment for this type

of cryptography. They are usually implemented with the group of points of an elliptic curve over a finite field, while for the pairing the most common choices are the Tate and Weil pairings, in their modified version so that $e(g, g) \neq 1$. For a detailed analysis of these groups and the curves to use in an implementation see [59]. The most used and studied types of bilinear groups are the ones with prime order p.

Definition 2.1 (Pairing)**.** Let $\mathbb{G}_1, \mathbb{G}_2$ be groups of the same prime order p. A symmetric pairing is a bilinear map e such that $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ has the following properties:

- Bilinearity: $\forall g, h \in \mathbb{G}_1, \forall a, b \in \mathbb{Z}_p, \quad e(g^a, h^b) = e(g, h)^{ab}$.
- Non-degeneracy: for g generator of $\mathbb{G}_1$, $e(g, g) \neq 1_{\mathbb{G}_2}$.

Definition 2.2 (Bilinear Group)**.** $\mathbb{G}_1$ is a Bilinear group if the conditions above hold and both the group operations in $\mathbb{G}_1$ and $\mathbb{G}_2$ as well as the bilinear map e are efficiently computable.

In the remainder of this section $\mathbb{G}_1$ and $\mathbb{G}_2$ are understood.

2.1.2 Security Assumptions on Prime Order Bilinear Groups

Here are presented the various assumptions on bilinear groups that will be used in the security proofs presented in this thesis. They are all variations of the bilinear Diffie-Hellman assumption, which is the equivalent of the Diffie-Hellman assumption for bilinear groups.

Decisional Bilinear Diffie-Hellman Assumption

The Decisional Bilinear Diffie-Hellman (BDH) assumption is the basilar assumption used for proofs of indistinguishability in pairing-based cryptography. It has been first introduced in [23] by Boneh and Franklin and then widely used in a variety of proofs, including the one of the first concrete ABE scheme in [38]. It is defined as follows.

Let $a, b, s, z \in \mathbb{Z}_p$ be chosen at random and g be a generator of the bilinear group $\mathbb{G}_1$. The decisional Bilinear Diffie-Hellman (BDH) problem consists in constructing an algorithm $\mathcal{B}(A = g^a, B = g^b, S = g^s, T) \to \{0, 1\}$ to efficiently distinguish between the tuples $(A, B, S, e(g, g)^{abs})$ and $(A, B, S, e(g, g)^z)$ outputting respectively 1 and 0. The advantage of $\mathcal{B}$ in this case is clearly written as:

$$Adv_{\mathcal{B}} = \left| \Pr\left[\mathcal{B}(A, B, S, e(g, g)^{abs}) = 1\right] - \Pr\left[\mathcal{B}(A, B, S, e(g, g)^z) = 1\right] \right|$$

where the probability is taken over the random choice of the generator g, of a, b, s, z in $\mathbb{Z}_p$, and the random bits possibly consumed by $\mathcal{B}$ to compute the response.

Definition 2.3 (BDH Assumption)**.** The decisional BDH assumption holds if no probabilistic polynomial-time algorithm $\mathcal{B}$ has a non-negligible advantage in solving the decisional BDH problem.

Augment Decisional Bilinear Diffie-Hellman Assumption

This assumption, introduced by Liang et al. in [51], is a variant of the basic BDH in which the attacker knows one more element, so is slightly stronger. We formally define it as follows.

Let $a, b, s,\ z \in \mathbb{Z}_p$ be exponents chosen at random, let g be a generator of the bilinear group $\mathbb{G}_1$, and let $b \neq 0$. The Augment Decisional Bilinear Diffie-Hellman (ABDH) problem consists in constructing an efficient algorithm $\mathcal{B}(A = g^a, B = g^b, C = g^{\frac{1}{b}}, S = g^s, Z) \to \{0, 1\}$ to distinguish between the tuples $(A, B, C, S, e(g, g)^{abs})$ and $(A, B, C, S, e(g, g)^z)$. The advantage of $\mathcal{B}$ is defined, following the standard convention, as:

$$Adv_{\mathcal{B}} = \left| \Pr\left[\mathcal{B}(A, B, C, S, e(g, g)^{abs}) = 1\right] - \Pr\left[\mathcal{B}(A, B, C, S, e(g, g)^z) = 1\right] \right|$$

where the probability is taken over the random choice of the generator g, of a, b, s, z in $\mathbb{Z}_p$, and the random bits possibly consumed by $\mathcal{B}$ to compute the response.

Definition 2.4 (ABDH Assumption)**.** The decisional ABDH assumption holds if no probabilistic polynomial-time algorithm $\mathcal{B}$ has a non-negligible advantage in solving the decisional ABDH problem.

In the next Section 2.1.3 we show an adaptation of these assumption to the generic group model and we are able to prove a related security bound.

2.1.3 Generic Security of Diffie-Hellman Assumptions

In [22] Boneh et. al. stated and proved a theorem that gives a lower bound on the advantage of a generic algorithm in solving a class of decisional Diffie-Hellman problems. Despite a lower bound in generic groups does not imply a lower bound in any specific group, it still provides evidence of soundness of the assumptions. In this section, first the general Diffie-Hellman Exponent Problem will be defined, then the lower bound will be stated, and finally it will be shown how the problems introduced in Section 2.1.2 may be seen as particular cases of the general problem.

General Diffie-Hellman Exponent Problem

Let p be a prime and let s, n be positive integers. Let $P, Q \in \mathbb{Z}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{Z}_p$ and let $f \in \mathbb{Z}_p[X_1, \ldots, X_n]$. Let $P = (p_1, p_2, \ldots, p_s)$ and $Q = (q_1, q_2, \ldots, q_s)$, we require that $p_1 = q_1 = 1$. Moreover define:

$$P(x_1, \ldots, x_n) = (p_1(x_1, \ldots, x_n), \ldots, p_s(x_1, \ldots, x_n)) \in (\mathbb{Z}_p)^s.$$

And similarly for the s-tuple Q. Let $\mathbb{G}_1, \mathbb{G}_2$ be groups of order p and let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a non-degenerate bilinear map. Let $g \in \mathbb{G}_1$ be a generator of $\mathbb{G}_1$ and set $g_2 = e(g, g) \in \mathbb{G}_2$. Let

$$H(x_1, \ldots, x_n) = \left(g^{P(x_1, \ldots, x_n)}, g_2^{Q(x_1, \ldots, x_n)}\right) \in \mathbb{G}_1^s \times \mathbb{G}_2^s,$$

we say that an algorithm $\mathcal{B}$ that outputs $b \in \{0,1\}$ has advantage ϵ in solving the Decision (P,Q,f)-Diffie-Hellman problem in $\mathbb{G}_1$ if

$$\left| Pr\left[\mathcal{B}\left(H(x_1,\ldots,x_n), g_2^{f(x_1,\ldots,x_n)}\right) = 0\right] - Pr\left[\mathcal{B}(H(x_1,\ldots,x_n),T) = 0\right]\right| > \epsilon$$

where the probability is over the random choice of generator $g \in \mathbb{G}_1$, the random choice of $x_1,\ldots,x_n$ in $\mathbb{Z}_p$, the random choice of $T \in \mathbb{G}_2$, and the random bits consumed by $\mathcal{B}$.

Definition 2.5 (Dependence on (P,Q))**.** Let $P,Q \in \mathbb{Z}_p[X_1,\ldots,X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{Z}_p$. We say that a polynomial $f \in \mathbb{Z}_p[X_1,\ldots,X_n]$ is dependent on the sets (P,Q) if there exist s^2+s constants $\{a_{i,j}\}_{i,j=1}^s, \{b_k\}_{k=1}^s$ such that

$$f = \sum_{i,j=1}^{s} a_{i,j} p_i p_j + \sum_{k=1}^{s} b_k q_k$$

We say that f is independent of (P,Q) if f is not dependent on (P,Q).

For a polynomial $f \in \mathbb{Z}_p[X_1,\ldots,X_n]^s$, let d_f denote the total degree of f. For a set $P \subseteq \mathbb{Z}_p[X_1,\ldots,X_n]^s$ let $d_P = \max\{d_f : f \in P\}$.

Complexity Lower Bound in Generic Bilinear Groups

We state the following lower bound in the framework of the generic group model. Consider two random encodings ξ_0, ξ_1 of the additive group $\mathbb{Z}_p$, i.e. injective maps $\xi_0, \xi_1 : \mathbb{Z}_p \to \{0,1\}^m$. For $i = 0,1$ write $\mathbb{G}_i = \{\xi_i(x) : x \in \mathbb{Z}_p\}$. We are given oracles to compute the induced group action on $\mathbb{G}_1, \mathbb{G}_2$, and an oracle to compute a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. We refer to $\mathbb{G}_1$ as a *generic* bilinear group. The following theorem gives a lower bound on the advantage of a generic algorithm in solving the decision (P,Q,f)-Diffie-Hellman problem. We emphasize, however, that a lower bound in generic groups does not imply a lower bound in any specific group.

Theorem 2.1 (Theorem A.2 of [22])**.** *Let $P,Q \in \mathbb{Z}_p[X_1,\ldots,X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{Z}_p$ and let $f \in \mathbb{Z}_p[X_1,\ldots,X_n]$. Let $d = \max(2d_P, d_Q, d_f)$. Let ξ_0, ξ_1 and $\mathbb{G}_1, \mathbb{G}_2$ be defined as above. If f is independent of (P,Q) then for any $\mathcal{A}$ that makes a total of at most q queries to the oracles computing the group operation in $\mathbb{G}_1, \mathbb{G}_2$ and the bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ we have:*

$$\left| Pr\left[\mathcal{A}\left(p, \xi_0(P(x_1,\ldots,x_n)), \xi_1(Q(x_1,\ldots,x_n)), \xi_1(t_0), \xi_1(t_1)\right) = b\right] - \frac{1}{2}\right| \leq \frac{(q+2s+2)^2 d}{2p}$$

Where $x_1,\ldots,x_n, y$ are chosen uniformly at random from $\mathbb{Z}_p$, b is chosen uniformly at random from $\{0,1\}$ and $t_b = f(x_1,\ldots,x_n), t_{1-b} = y$.

Corollary 2.1 (Corollary A.3 of [22])**.** *Let $P,Q \in \mathbb{Z}_p[X_1,\ldots,X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{Z}_p$ and let $f \in \mathbb{Z}_p[X_1,\ldots,X_n]$. Let $d = \max(2d_P, d_Q, d_f)$. If f is independent of (P,Q) then any $\mathcal{A}$ that has advantage $\frac{1}{2}$ in solving the decision (P,Q,f)-Diffie-Hellman Problem in a generic bilinear group G must take time at least $\Omega(\frac{p}{d} - s)$.*

Using Corollary 2.1

We claim that the assumptions presented in Section 2.1.2 follow from Corollary 2.1 giving the sets P, Q that reduces them to the general bilinear Diffie-Hellman problem:

- BDH in $\mathbb{G}_1$: set $P = \{1, y, w, z\}, Q = \{1\}, f = ywz$.
- ABDH in $\mathbb{G}_1$: set $P = \{1, y, w, \frac{1}{w}, z\}, Q = \{1\}$,$f = y\ w\ z$.

It is easy to see that each f is independent to the respective sets P and Q, in fact multiplying any two polynomials in the sets P and then combining them linearly does not give the polynomial f. To see this explicitly in the case of ABDH, the complete list of terms that may be obtained combining any two polynomials of P follows:

$$1, w, \frac{1}{w}, y, yw, \frac{y}{w}, wz, \frac{z}{w}, z, yz$$

Since there is no monomial in which y, w, and z appear together, it is apparent that no linear combination of these terms may give ywz as result, thus f is independent of P, Q.

Thus applying the Corollary 2.1 a lower bound on the computational complexity of these problems in the generic bilinear group is obtained.

2.1.4 Interactive Diffie-Hellman

Interactive assumptions are usually stronger than their static counterparts, since the solver has more control over the parameters. This unfortunately means that they give weaker security, but sometimes it is nearly impossible to reduce highly interactive protocols to static assumptions, so interactive assumptions are needed.

Interactive Decisional Diffie-Hellman Assumption

Let $\mathcal{C}$ be a challenger that chooses $a, b, z \in \mathbb{Z}_p$ at random and g be a generator of a group $\mathbb{G}$ of prime order p. The *Interactive Decisional Diffie-Hellman* (IBDDH) problem consists in constructing an algorithm $\mathcal{B}(\mathcal{C}) \to \{0, 1\}$ that interacts with the challenger in the following way:

- $\mathcal{C}$ gives to $\mathcal{B}$ the values $A = g^a, B = g^b$;
- $\mathcal{B}$ chooses an exponent $0 \neq s \in \mathbb{Z}_p$ and sends to the challenger $S = B^{\frac{1}{s}}$;
- $\mathcal{C}$ flips a random coin $r \in \{0, 1\}$ and answers with $Z = S^a = g^{\frac{ab}{s}}$ if $r = 0$, $Z = g^z$ if $r = 1$;
- $\mathcal{B}$ given A, B, S, Z outputs a guess r' of r.

The advantage of $\mathcal{B}$ in this case is clearly written as:

$$Adv_{\mathcal{B}} = \left| \Pr\left[\mathcal{B}(A, B, S, g^{\frac{ab}{s}}) = 0\right] - \Pr\left[\mathcal{B}(A, B, S, g^z) = 0\right] \right|$$

where the probability is taken over the random choice of the generator g, of a, b, s, z in $\mathbb{Z}_p$, of $r \in \{0, 1\}$, and the random bits possibly consumed by $\mathcal{B}$ to compute the response.

Definition 2.6 (IDDH Assumption)**.** The Interactive Decisional DH assumption holds if no probabilistic polynomial-time algorithm $\mathcal{B}$ has a non-negligible advantage in solving the decisional IDDH problem.

Note that an adversary that can solve the IDDH problem can solve the DH problem simulating a IDDH problem and choosing $s = 1$, but the converse is not true since it is not possible to adapt the DH challenge without knowing s.

2.2 Security of Cryptographic Primitives

Many high-level protocols use cryptographic primitives (such as ciphers) as building blocks. Unfortunately the security of many of these primitives does not directly derive from the hardness of a well known problem, so the assumption becomes that these core components satisfy some fundamental properties.

In this section will be presented the main characteristics of the primitives used in the protocols presented in this thesis, alongside the assumptions on their security.

2.2.1 Security of Digital Signatures and ECDSA

To validate an action having a legal value we are usually requested to produce our handwritten signature. Assuming that nobody is able to forge a signature while anybody can verify its validity, this method is used to certify the correspondence of the identities of who is taking the action and in the name of whom the action is being taken.

In the digital word, handwritten signatures are substituted by *digital signatures* that satisfy the same conditions seen above. With the name *Digital Signature Scheme* we refer to any asymmetric cryptographic scheme for producing and verifying digital signatures.

A Digital Signature Scheme consists of three algorithms:

- *Key Generation* - $\mathsf{KeyGen}(\kappa) \rightarrow (\mathsf{SK}, \mathsf{PK})$: given a security parameter κ generates a public key PK, that is published, and a secret key SK.

- *Signing* - $\mathsf{Sign}(m, \mathsf{SK}) \rightarrow s$: given a message m and the secret key SK, computes a digital signature s of m.

- *Verifying* - $\mathsf{Ver}(m, s, \mathsf{PK}) \rightarrow r$: given a message m, a signature s and the public key PK, it outputs the result $r \in \{\mathsf{True}, \mathsf{False}\}$ that says whether or not s is a valid signature of m computed by the secret key corresponding to PK.

The previous algorithms usually require a source of random bits to operate securely.

A common security requirement for a Digital Signature Scheme is the difficulty of forging a signature, modelled by the following game (commonly known as an *existential forgery*).

Definition 2.7 (Digital Signature Security Game)**.** Let $\mathcal{DSS}$ be a Digital Signature Scheme. Its security game, for an adversary $\mathcal{A}$, proceeds as follows:

Setup. The challenger $\mathcal{C}$ runs the KeyGen algorithm, and gives to the adversary the public key PK.

Query. The adversary issues signature queries for some messages m_i, the challenger answers giving $s_i = \mathsf{Sign}(m_i, \mathsf{SK})$.

Challenge. The adversary selects a message m such that $m \neq m_i \ \forall i$, and tries to compute a forged signature s. $\mathcal{A}$ wins if $\mathsf{Ver}(m, s, \mathsf{PK}) = \mathsf{True}$.

Definition 2.8 (Security of a Digital Signature Scheme)**.** A Digital Signature Scheme $\mathcal{DSS}$ is said *secure* if there is no polynomial-time algorithm $\mathcal{A}$ (w.r.t. κ) that wins the Digital Signature Security Game 2.7 with non-negligible probability.

Ideally, a Digital Signature Scheme is designed in such a way that forging a signature in the scheme is equivalent to solving a hard mathematical problem Although this equivalence is usually assumed but not proved, we say that the Digital Signature Scheme is based on that mathematical problem. Several Digital Signatures Schemes (e.g. [33]), are based on the discrete logarithm problem (although other approaches exist, see e.g. [71], [72]). Among them, the Elliptic Curve Digital Signature Algorithm (ECDSA), which uses elliptic curves, is widespread. To the sake of easy reference we recall briefly how ECDSA is designed [43].

Domain Parameters An elliptic curve $\mathbb{E}$ defined over a finite field $\mathbb{F}_q$ is fixed together with a rational point $P \in \mathbb{E}(\mathbb{F}_q)$ having order n, and a cryptographic hash function h [70]. Let $\mathcal{O}$ be the point at infinity of $\mathbb{E}$.

Key Generation Any user A selects a random integer d in the interval $[2, n-1]$. Then his *public key* is $Q = dP$ while d is his *private key.*

Signing To sign a message m (a binary string), a user A with key pair (Q, d) selects a random integer k in the interval $[2, n-1]$ and accepts it if it is relatively prime to n. Then A computes $(x_1, y_1) = kP$ and converts x_1 in an integer $\bar{x}_1$. In the unlikely event that $r = \bar{x}_1 \mod n$ is 0, then another random integer k has to be extracted. Otherwise, A proceeds to hash the message and to transform the digest $h(m)$ in a non-negative integer e using the standard conversion of the binary representation. Then A computes the value $s = k^{-1}(e + dr) \mod n$. In the unlikely event that $s = 0$, the value of k must be randomly selected again, else the pair (r, s) is output as A's signature for the message m.

Verifying To verify the signature (r, s) with the public key Q for the message m a user B proceeds as follows: first r and s are checked to be integers contained in the interval $[1, n-1]$. Then B computes the hash of the message $h(m)$ and

converts it to a non-negative integer e. Then the point of the elliptic curve $(x, y) = (es^{-1} \mod n)P + (rs^{-1} \mod n)Q$ is computed. If $Q = \mathcal{O}$, then B rejects the signature; otherwise x is converted into an integer $\bar{x}$ and the signature is accepted if and only if $\bar{x} = r \mod n$.

Note that if the signature (r, s) of the message m was actually produced by the private key d, then $s = k^{-1}(e + dr) \mod n$ and so:

$$k = s^{-1}(e + dr) = es^{-1} + drs^{-1} \mod n$$

and

$$kP = (x_1, y_1) = \left(es^{-1} + drs^{-1} \mod n\right) P.$$

Clearly, if an attacker is able to solve the DLOG on $\mathbb{E}$, then this adversary can break the corresponding ECDSA. The converse is much less obvious. In [68], the authors provide convincing evidence that the unforgeability of several discrete logarithm-based signatures cannot be equivalent to the discrete logarithm problem in the standard model. Their impossibility proofs apply to many discrete logarithm-based signatures like ElGamal signatures and their extensions, DSA, ECDSA and KCDSA, as well as standard generalizations of these. However, their work does not explicitly lead to actual attacks. Assuming that breaking the DLOG is the most efficient attack on ECDSA, then nowadays recommended key lengths start from 160 bits, with 256 bits being the length of the signatures employed in the Bitcoin protocol.

2.2.2 Security of Hash Functions

Hash functions are somewhat the digital equivalent of a fingerprint: they compress a file in a digest that represents it, so that (ideally) no two different files correspond to the same digest. Moreover it should be almost impossible to reconstruct the whole file from the digest, but at the same time it is easy to show that a given file corresponds to a given digest.

A hash function H can be idealized as a function whose set of inputs is the set of all possible binary strings, denoted by $(\mathbb{F}_2)^*$, while its set of possible outputs is the set of all binary strings of given length (called *digest*). Real-life hash functions have a *finite* input set, but it is so large that can be thought of as infinite. For example, the hash functions used in the Bitcoin protocol are SHA256, enjoying a digest length of 256 bits and with input string up to 2^{64}-bit long, and RIPEMD-160, with 160-bit digests.

Cryptographic hash functions can need several security assumptions, however for the proofs presented in this thesis the following definitions are sufficient.

Definition 2.9 (Collision Problem for a Class of Inputs)**.** Let $r \geq 1$. Let $h : (\mathbb{F}_2)^* \to (\mathbb{F}_2)^r$ be a *hash function*, and $L \subseteq (\mathbb{F}_2)^l$ be a class of inputs. The *collision problem* for h and L consists in finding two different inputs $m_1, m_2 \in L$, with $m_1 \neq m_2$, such that $h(m_1) = h(m_2)$.

Definition 2.10 (Collision Resistance of Hash Functions)**.** Let h be a *hash function*. We say that h is *collision resistant* for a class of inputs L if there is no polynomial-time algorithm $\mathcal{B}(h, L) \to \{m_1, m_2\}$ that solves the Collision Problem 2.9 for h and L with non-negligible probability. The complexity parameter is the length of inputs l.

For example, as shown in [66], SHA-256 has passed several statistical tests designed to verify "the absence of any detectable correlation between input and output, and the absence of any detectable bias due to single bit changes in the input string". Therefore, it can be considered collision-resistant.

2.2.3 Security of Symmetric Ciphers

Symmetric ciphers are the most important cryptographic primitive and the most widely used in real-world applications. They provide a fast way to scramble data so that only who possess a copy of the same key used to alter the message can get any information out of it. In particular these ciphers are designed to be extremely fast and to use very few resources, so they can be embedded in every kind of device. This is the reason why in most protocols asymmetric encryption is used to exchange the keys so symmetric ciphers can be used to actually encrypt the bulk of data.

Unfortunately the design of these ciphers is not based on hard mathematical problems, but instead they try to propagate the randomness of the key in order to simulate a random function that would completely hide any information in the output. So it is not feasible to prove their security, but it is only possible to assess the randomness of the output with statistical tests and control that no known cryptanalysis technique is effective.

For this reason their security has to be modelled and taken as an assumption in order to prove the security of protocols that use symmetric ciphers as building blocks.

For the proof of the tokenization algorithm presented in Chapter 4 we assume that the cipher possess the *Indistinguishability under Chosen Plaintext Attack* (IND-CPA) security property, that is formally defined as follows.

Definition 2.11 (IND-CPA)**.** Let $E(K, m) \to c$ be an encrypting function. An *Indistiguishability under Chosen Plaintext Attack* (IND-CPA) game for E between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ proceeds as follows:

Phase I $\mathcal{A}$ chooses a plaintext m_i and sends it to $\mathcal{C}$, that responds with $c_i = E(K, m_i)$. This phase is repeated a polynomial number of times.

Challenge $\mathcal{A}$ chooses two plaintexts m_0^*, m_1^* (never chosen in Phase I) and sends them to $\mathcal{C}$, that selects $\nu \in \{0, 1\}$ at random and computes $c = E(K, m_\nu^*)$. Then $\mathcal{C}$ sends c to $\mathcal{A}$.

Phase II Phase I is repeated (a polynomial number of times), with the obvious restriction that $\mathcal{A}$ cannot choose m_0^* or m_1^*.

Guess $\mathcal{A}$ guesses $\nu' \in \{0, 1\}$, and wins if $\nu' = \nu$.

We say that the advantage $Adv_{\mathcal{A}}^{E}$ of $\mathcal{A}$ winning the IND-CPA game for the encrypting algorithm E is:

$$Adv_{\mathcal{A}}^{E} = \left| Pr\left[\nu' = \nu\right] - \frac{1}{2} \right|$$

E is said to be *secure in a IND-CPA scenario* if there is no probabilistic polynomial-time algorithm $\mathcal{A}$ that wins the CPA game with more than negligible advantage.

For example, in [75] the authors showed that AES-256 passed statistical tests designed to verify the following properties:

- "the absence of any detectable correlation between plaintext/ciphertext pairs and the absence of any detectable bias due to single bit changes to a plaintext block";
- "the absence of any detectable deviations from randomness".

Therefore AES-256 can be considered to be IND-CPA secure.

Chapter 3

Multi-Authority Key-Policy Attribute Based Encryption

Attribute-based encryption (ABE) addresses the problem of access control over encrypted data stored in the cloud, providing efficient and expressive methods to protect shared data while maintaining the control on the accesses on a fine-grained level. A multitude of different ABE schemes have been proposed, designed to address various requirements. In particular there are a few systems that use multiple authorities, but limited to the ciphertext-policy instance. In this chapter two multi-authority key-policy ABE schemes will be shown.

This original work has been presented at the CAI conference: the first scheme at the 6th edition held in Stuttgart, in September 2015, and published in the proceedings [55]; the second scheme [56] at the 7h edition held in Kalamata, in June 2017 and is under review for a publication in a special issue of the journal *Theoretical Computer Science*. It is a joint work with Dr. Chiara Marcolla and Prof. Massimiliano Sala. The graduand designed the protocols and proved their security, while the co-authors helped refining the proofs and the paper and gave the seminal idea for the collaborative variant of the protocol.

3.1 Cryptography for the Cloud

The key feature that makes the cloud so attracting nowadays is the great accessibility it provides: users can access their data through the Internet from anywhere. Unfortunately, at the moment the protection offered for sensitive information is questionable and access control is one of the greatest concerns. Illegal access may come from outsiders or even from insiders without proper clearance. One possible approach for this problem is to use Attribute-Based Encryption (ABE) that provides cryptographically enhanced access control functionality to encrypted data.

In Key Policy ABE (KP-ABE), each ciphertext is described by some attributes (e.g. competence area), and for each user in the system a private key is issued by an authority. This key specifies what it can access to, in the form of a boolean formula over said attributes, reflecting the credential of its carrier.

A user will be able to decrypt a ciphertext if the attributes associated with this ciphertext satisfy the boolean formula associated to the user's private key.

ABE developed from Identity Based Encryption, a scheme proposed by Shamir [77] in 1985 with the first constructions obtained in 2001 by Boneh and Franklin [23]. The use of bilinear groups, in particular the Tate and Weil pairings on elliptic curves [23], was the winning strategy that finally allowed to build schemes following the seminal idea of Shamir. Bilinear groups came in nicely when a preliminary version of ABE was invented by Sahai and Waters [76] in 2005. Immediately afterwards, Goyal, Pandey, Sahai, and Waters [38] formulated the two complementary forms of ABE that are nowadays standard: ciphertext-policy ABE and key-policy ABE. In a ciphertext-policy ABE system, keys are associated with sets of attributes and ciphertexts are associated with access policies. In a KP-ABE system, the situation is reversed: keys are associated with access policies and ciphertexts are associated with sets of attributes.

Several developments in efficiency and generalizations have been obtained for key-policy ABE, e.g. [7], [41], [67]. A first implementation of ciphertext-policy ABE has been achieved by Bethencourt et al. [18] in 2007 but the proofs of security of the ciphertext-policy ABE remained unsatisfactory since they were based on an assumption independent of the algebraic structure of the group (the generic group model). It is only with the work of Waters [85] that the first non-restricted ciphertext-policy ABE scheme was built with a security based on variations of the DH assumption for bilinear groups.

Related to the schemes presented here is the construction for multiple authorities (ciphertext-policy ABE) that have been proposed in [26], [27] and [50]. The approach of CP-ABE to multiple authorities is quite different, since the decentralization is not meant as a form of strengthening the control over accesses, but rather as an effort to distribute the control over the attributes between multiple authorities. In fact in CP-ABE users are described by a set of attributes, so in a decentralized environment different authorities are in charge of assigning (or not) attributes out of a specific subset. The main challenge in this case is to prevent collusion (that is effectively combining attributes of different users) without a central authority.

Original constructions In this chapter two multi-authority KP-ABE schemes are presented. In the first system, after the creation of an initial set of common parameters, the authorities may be set up in any moment and without any coordination. A party can simply act as an ABE authority by creating public parameters and issuing private keys to different users (assigning access policies while doing so). A user can encrypt data under any set of attributes specifying also a set of *trusted* authorities, so the encryptor maintains high control. Also, the system does not require any central authority. This scheme has very short single-authority keys, that compensate the need of multiple keys (one for authority). Moreover, the pairing computations in the bilinear group are involved only during the decryption phase, thus obtaining significant advantages in terms of encryption times. Even if the authorities are collaborating, the existence of just one non-cheating authority guarantees that no illegitimate party (including authorities) has access to the encrypted data. The scheme is proven secure using the classical bilinear Diffie-Hellman assumption.

The second scheme aims to improve the efficiency of the first one, and in particular to reduce the size of the keys and public parameters that reflect in much shorter ciphertext and a great speed up of decryption. To achieve this goal a round of collaboration between authorities is introduced. This means that each authority processes the public parameters so a single set is required and the individual private keys given to a user can be combined into one key. So at the cost of less flexibility in the choice of authorities (every single one is required to decrypt) there is a significant efficiency improvement. This scheme is proven secure using a variation of the bilinear Diffie-Hellman assumption.

Organization This chapter is organized as follows. In Section 3.2 the main mathematical tools used in the construction of the multi authority KP-ABE schemes are presented. In Section 3.3 the first multi authority KP-ABE scheme is explained in detail and its security is proven under standard, non-interactive assumptions in the selective set model. In Section 3.4 the second scheme is presented in full details, complete with the proof of security.

3.2 Background: Access Structures and Linear Secret Sharing Schemes

In this section the definition and main properties of two fundamental blocks of ABE schemes are reported: access structures and linear secret sharing schemes. These tools are used to model the access policies and embed them in the encryption process in the vast majority of ABE constructions. See the cited references for more details on these topics.

Access structures define who may and who may not access the data, listing the sets of attributes that have clearance.

Definition 3.1 (Access Structure)**.** An access structure $\mathbb{A}$ on a universe of attributes U is the set of the subsets $S \subseteq U$ that are authorized. That is, a set of attributes S satisfies the policy described by the access structure $\mathbb{A}$ if and only if $S \in \mathbb{A}$.

Access structures are used to describe a policy of access, that is the rules that prescribe who may and may not access the information. If these rules are constructed using only AND, OR and THRESHOLD operators on the attributes (a (k, n) threshold operator specifies that k attributes are requested out of a set of n specific attributes), then the access structure is *monotonic*.

Definition 3.2 (Monotonic Access Structure)**.** An access structure $\mathbb{A}$ is said monotonic if given $S_0 \subseteq S_1 \subseteq U$ it holds

$$S_0 \in \mathbb{A} \Longrightarrow S_1 \in \mathbb{A}$$

An interesting property is that monotonic access structures may be converted into linear secret sharing schemes (LSSS). In this setting a secret is divided in *shares* distributed between the parties of the LSSS, that are identified with the attributes of the access structure. Then an access policy is satisfied if there are enough parties to allow the reconstruction of the secret using the corresponding

shares. This is equivalent to having enough attributes, note that the monotonic property means that extra attributes do not jeopardize the satisfaction of the policy.

A LSSS may be defined as follows (adapted from [13]).

Definition 3.3 (Linear Secret-Sharing Schemes (LSSS))**.** A secret-sharing scheme Π over a set of parties P is called linear (over $\mathbb{Z}_p$) if

1. The shares for each party form a vector over $\mathbb{Z}_p$.
2. There exists a matrix M with l rows and n columns called the share-generating matrix for Π. For all $i \in \{1, \ldots, l\}$ M_i (the i-th row of M) is labeled via a function ρ, that associates it to the party $\rho(i)$. Considering the vector $\vec{v} = (s, r_2, \ldots, r_n) \in \mathbb{Z}_p^n$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_i \in \mathbb{Z}_p$, with $i \in \{2, \ldots, n\}$ are randomly chosen, then $M\vec{v}$ is the vector of l shares of the secret s according to Π. The share $(M\vec{v})_i = M_i\vec{v}$ belongs to party $\rho(i)$.

It is shown in [13] that every linear secret sharing-scheme according to the above definition also enjoys the linear reconstruction property, defined as follows: suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subseteq \{1, \ldots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $w_i \in \mathbb{Z}_p$, with $i \in I$ such that, if λ_i are valid shares of any secret s according to Π, then

$$\sum_{i \in I} w_i \lambda_i = s \tag{3.1}$$

Furthermore, it is shown in [13] that these constants w_i can be found in time polynomial in the size of the share-generating matrix M.

Note that the vector $(1, 0, \ldots, 0)$ is the "target" vector for the linear secret sharing scheme. Then, for any set of rows I in M, the target vector is in the span of I if and only if I is an authorized set. This means that if I is not authorized, then for any choice of $c \in \mathbb{Z}_p$ there will exist a vector $\vec{u}$ such that $u_1 = c$ and

$$Mi \cdot \vec{u} = 0 \qquad \forall i \in I \tag{3.2}$$

In the first ABE schemes the access formulas are typically described in terms of access trees, that represent boolean formulas on the attributes in a very convenient and readable way. In the appendix of [50] a method to perform a conversion from access trees to LSSS is presented, in [54] there is a more comprehensive discussion on access tree conversion, and the authors propose an algorithm that generates much smaller matrices which allow more efficiency in ABE implementations.

See [38], [13] and for more details about LSSS and access structures.

3.3 Multi-Authority Key-Policy Attribute-Based Encryption

This section is divided in three parts. First the definitions of Multi-Authority Key-Policy ABE and of CPA selective security are presented. In the second part

the first scheme is presented in detail and, finally, the security of this scheme is proven under the classical BDH assumption in the selective set model.

A security parameter will be used to determine the size of the bilinear group used in the construction, and therefore the order of complexity of the security assumption. In practice first the parameter is chosen to achieve a certain security level, then this value is used to compute the order that the bilinear group must have, finally a suitable group is picked and used.

3.3.1 Multi Authority KP-ABE Structure and Security

In this scheme, after the common universe of attributes and bilinear group are agreed, the authorities set up independently their master key and public parameters. The master key is subsequently used to generate the private keys requested by users. Users ask an authority for keys that embed a specific access structure, and the authority issues the key only if it judges that the access structure suits the user that requested it. Equivalently an authority evaluates a user that requests a key, assigns an access structure, and gives to the user a key that embeds it.

When someone wants to encrypt, it chooses a set of attributes that describes the message (and thus determines which access structures may read it) and a set of trusted authorities. The ciphertext is computed using the public parameters of the chosen authorities, and may be decrypted only using a valid key for each of these authorities. A key with embedded access structure $\mathbb{A}$ may be used to decrypt a ciphertext that specifies a set of attributes S if and only if $S \in \mathbb{A}$, that is the structure considers the set authorized.

Security Game

This scheme is secure under the classical BDH assumption in the selective set model, in terms of chosen-plaintext indistinguishability. The security game is formally defined as follows.

Let $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt})$ be a MA-KP-ABE scheme for a message space $\mathcal{M}$, a universe of authorities X and an access structure space $\mathcal{G}$ and consider the following MA-KP-ABE experiment $\mathsf{MA\text{-}KP\text{-}ABE\text{-}Exp}_{\mathcal{A},\mathcal{E}}(\lambda, U)$ for an adversary $\mathcal{A}$, security parameter λ, and attribute universe U:

Init. The adversary declares the set of attributes S and the set of authorities $A \subseteq X$ that it wishes to be challenged upon. Moreover it selects the *honest authority* $k_0 \in A$.

Setup. The challenger runs the Setup algorithm, initializes the authorities and gives to the adversary the public parameters.

Phase I. The adversary issues queries for private keys of any authority, but k_0 answers only to queries that ask for keys related to access structures $\mathbb{A}_i$ such that $S \notin \mathbb{A}_i \ \forall i$. On the contrary the other authorities respond to every query.

Challenge. The adversary submits two equal length messages m_0 and m_1. The challenger flips a random coin $b \in \{0, 1\}$, and encrypts m_b with S for the set of authorities A. The ciphertext is passed to the adversary.

Phase II. Phase I is repeated.

Guess. The adversary outputs a guess b' of b.

The experiment has result 1 if $b' = b$, 0 otherwise.

Definition 3.4 (MA-KP-ABE Selective Security)**.** The MA-KP-ABE scheme $\mathcal{E}$ is CPA selective secure (or secure against chosen-plaintext attacks) for attribute universe U if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function $negl$ such that:

$$\Pr[\mathsf{MA\text{-}KP\text{-}ABE\text{-}Exp}_{\mathcal{A},\mathcal{E}}(\lambda, U) = 1] \leq \frac{1}{2} + negl(\lambda).$$

3.3.2 The Scheme

The scheme plans a set X of independent authorities, each with their own parameters, and it sets up an encryption algorithm that lets the encryptor choose a set $A \subseteq X$ of authorities, and combines their public parameters in such a way that an authorized key for each authority in A is required to successfully decrypt.

The scheme consists of three randomized algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}$) plus the decryption $\mathsf{Decrypt}$. The techniques used are inspired from the scheme of Goyal et al. in [38]. The scheme works in a bilinear group $\mathbb{G}_1$ of prime order p, and uses LSSS matrices to share secrets according to the various access structures. Attributes are seen as elements of $\mathbb{Z}_p$.

The description of the algorithms follows.

$\mathsf{Setup}(U, g, \mathbb{G}_1) \to (\mathsf{PK}_k, \mathsf{MK}_k)$. Given the universe of attributes U and a generator g of $\mathbb{G}_1$ each authority sets up independently its parameters. For $k \in X$ the Authority k chooses uniformly at random $\alpha_k \in \mathbb{Z}_p$, and $z_{k,i} \in \mathbb{Z}_p$ for each $i \in U$. Then the authority's public parameters PK_k and master key MK_k are:

$$\mathsf{PK}_k = (Y_k = e(g,g)^{\alpha_k}, \{T_{k,i} = g^{z_{k,i}}\}_{i \in U}) \tag{3.3}$$

$$\mathsf{MK}_k = (\alpha_k, \{z_{k,i}\}_{i \in U}) \tag{3.4}$$

$\mathsf{KeyGen}_k(\mathsf{MK}_k, (M_k, \rho_k)) \to \mathsf{SK}_k$. The key generation algorithm for the authority k takes as input the master secret key MK_k and an LSSS access structure (M_k, ρ_k), where M_k is an $l \times n$ matrix on $\mathbb{Z}_p$ and ρ_k is a function which associates rows of M_k to attributes. It chooses uniformly at random a vector $\vec{v}_k \in \mathbb{Z}_p^n$ such that $v_{k,1} = \alpha_k$. Then it computes the shares $\lambda_{k,i} = M_{k,i}\vec{v}_k$ for $1 \leq i \leq l$ where $M_{k,i}$ is the i-th row of M_k. Then the private key SK_k is:

$$\mathsf{SK}_k = \left\{ K_{k,i} = g^{\frac{\lambda_{k,i}}{z_{k,\rho_k(i)}}} \right\}_{1 \leq i \leq l} \tag{3.5}$$

$\mathsf{Encrypt}(m, S, \{\mathsf{PK}_k\}_{k \in A}) \to \mathsf{CT}$. The encryption algorithm takes as input the public parameters, a set S of attributes and a message m to encrypt. It chooses $s \in \mathbb{Z}_p$ uniformly at random and then computes the ciphertext as:

$$\mathsf{CT} = \left(S, C' = m \cdot \left(\prod_{k \in A} Y_k \right)^s, \{C_{k,i} = (T_{k,i})^s\}_{k \in A,\ i \in S} \right) \tag{3.6}$$

$\mathsf{Decrypt}(\mathsf{CT}, \{\mathsf{SK}_k\}_{k\in A}) \to m'$. The input is a ciphertext for a set of attributes S and a set of authorities A and an authorized key for every authority cited by the ciphertext. Let (M_k, ρ_k) be the LSSS associated to the key k, and suppose that S is authorized for each $k \in A$. Thanks to the linear reconstruction property (3.1), the algorithm finds $w_{k,i} \in \mathbb{Z}_p, i \in I_k$ for each $k \in A$ such that

$$\sum_{i\in I_k} \lambda_{k,i} w_{k,i} = \alpha_k \tag{3.7}$$

for appropriate subsets $I_k \subseteq S$ and then proceeds to reconstruct the original message computing:

$$\begin{aligned}
m' &= \frac{C'}{\prod_{k\in A}\prod_{i\in I_k} e(K_{k,i}, C_{k,\rho_k(i)})^{w_{k,i}}} \\
&= \frac{m \cdot \left(\prod_{k\in A} Y_k\right)^s}{\prod_{k\in A}\prod_{i\in I_k} e\left(g^{\frac{\lambda_{k,i}}{z_{k,\rho_k(i)}}}, (T_{k,\rho_k(i)})^s\right)^{w_{k,i}}} \\
&= \frac{m \cdot \left(\prod_{k\in A} e(g,g)^{\alpha_k}\right)^s}{\prod_{k\in A}\prod_{i\in I_k} e\left(g^{\frac{\lambda_{k,i}}{z_{k,\rho_k(i)}}}, (g^{z_{k,\rho_k(i)}})^s\right)^{w_{k,i}}} \\
&= \frac{m \cdot \left(\prod_{k\in A} e(g,g)^{\alpha_k}\right)^s}{\prod_{k\in A}\prod_{i\in I_k} e\left(g,g\right)^{\frac{\lambda_{k,i}}{z_{k,\rho_k(i)}} z_{k,\rho_k(i)} s w_{k,i}}} \\
&= \frac{m \cdot e(g,g)^{s(\sum_{k\in A}\alpha_k)}}{\prod_{k\in A} e(g,g)^{s\sum_{i\in I_k} w_{k,i}\lambda_{k,i}}} \\
&\overset{*}{=} \frac{m \cdot e(g,g)^{s(\sum_{k\in A}\alpha_k)}}{e(g,g)^{s(\sum_{k\in A}\alpha_k)}} = m
\end{aligned}$$

Where $\overset{*}{=}$ follows from property (3.7).

3.3.3 Security

The scheme is proved secure under the BDH assumption (Definition 2.3) in the selective set security game 3.3.1 in which every authority but one is supposed curious (or corrupted or breached) and then it will issue even keys that have enough clearance for the target set of attributes, while the honest one issues only unauthorized keys. Thus if at least one authority remains trustworthy the scheme is secure.
The security is provided by the following theorem.

Theorem 3.1. *If an adversary can break the scheme, then a simulator can be constructed to play the Decisional BDH game with a non-negligible advantage.*

Proof. Suppose there exists a polynomial-time adversary $\mathcal{A}$, that can attack the scheme in the Selective-Set model with advantage ϵ. Then a simulator $\mathcal{B}$ can be built that can play the Decisional BDH game with advantage $\epsilon/2$. The simulation proceeds as follows.

Init The simulator takes in a BDH challenge $(\vec{y} = (g, g^a, g^b, g^s), Z)$. The adversary chooses the challenge access structure S.

Setup The simulator chooses random $r_k \in \mathbb{Z}_p$ for $k \in A \setminus \{k_0\}$ and implicitly sets $\alpha_k = -r_k b$ for $k \in A \setminus \{k_0\}$ and $\alpha_{k_0} = ab + b\sum_{k \in A \setminus \{k_0\}} r_k$ by computing:

$$Y_{k_0} = e(g,g)^{\alpha_{k_0}} = e(g^a, g^b) \prod_{k \in A \setminus \{k_0\}} (g^b, g^{r_k})$$
$$Y_k = e(g,g)^{\alpha_k} = e(g^b, g^{-r_k}) \qquad \forall k \in A \setminus \{k_0\}$$

Then it chooses $z'_{k,i} \in \mathbb{Z}_p$ uniformly at random for each $i \in U$, $k \in A$ and implicitly sets

$$z_{k,i} = \begin{cases} z'_{k,i} & \text{if } i \in S \\ b z'_{k,i} & \text{if } i \notin S \end{cases}$$

Then it can publish the public parameters computing the remaining values as:

$$T_{k,i} = g^{z_{k,i}} = \begin{cases} g^{z'_{k,i}} & \text{if } i \in S \\ (g^b)^{z'_{k,i}} & \text{if } i \notin S \end{cases}$$

Phase I In this phase the simulator answers private key queries. For the queries made to the authority k_0 the simulator has to compute the $K_{k_0,i}$ values of a key for an access structure (M, ρ) with dimension $l \times n$ that is not satisfied by S. Therefore for the property 3.2 of the LSSS it can find a vector $\vec{u} \in \mathbb{Z}_p^n$ with $u_1 = 1$ fixed such that

$$M_i \vec{u} = 0 \qquad \forall i \text{ such that } \rho(i) \in S \tag{3.8}$$

Then it chooses uniformly at random a vector $\vec{v} \in \mathbb{Z}_p^n$ and implicitly sets the shares of $\alpha_{k_0} = b(a + \sum_{k \in A \setminus \{k_0\}} r_k)$ as

$$\lambda_{k_0,i} = b \sum_{j=1}^{n} M_{i,j}(v_j + (a + \sum_{k \in A \setminus \{k_0\}} r_k - v_1)u_j)$$

Note that $\lambda_{k_0,i} = \sum_{j=1}^{n} M_{i,j} w_j$ where $w_j = b(v_j + (a + \sum_{k \in A \setminus \{k_0\}} r_k - v_1)u_j)$ thus $w_1 = b(v_1 + (a + \sum_{k \in A \setminus \{k_0\}} r_k - v_1)1) = ab + b\sum_{k \in A \setminus \{k_0\}} r_k = \alpha_{k_0}$ so the shares are valid. Note also that from (3.8) it follows that

$$\lambda_{k_0,i} = b \sum_{j=1}^{n} M_{i,j} v_j \quad \forall i \text{ such that } \rho(i) \in S$$

Thus if i is such that $\rho(i) \in S$ the simulator can compute

$$K_{k_0,i} = (g^b)^{\frac{\sum_{j=1}^{n} M_{i,j} v_j}{z'_{k_0,\rho(i)}}} = g^{\frac{\lambda_{k_0,i}}{z_{k_0,\rho(i)}}}$$

Otherwise, if i is such that $\rho(i) \notin S$ the simulator computes

$$\begin{aligned} K_{k_0,i} &= g^{\frac{\sum_{j=1}^{n} M_{i,j}(v_j + (\sum_{k \in A \setminus \{k_0\}} r_k - v_1)u_j)}{z'_{k_0,\rho(i)}}} (g^a)^{\frac{\sum_{j=1}^{n} M_{i,j} u_j}{z'_{k_0,\rho(i)}}} \\ &= g^{\frac{b\sum_{j=1}^{n} M_{i,j}(v_j + (\sum_{k \in A \setminus \{k_0\}} r_k - v_1)u_j)}{bz'_{k_0,\rho(i)}}} g^{\frac{ba\sum_{j=1}^{n} M_{i,j} u_j}{bz'_{k_0,\rho(i)}}} \\ &= g^{\frac{\sum_{j=1}^{n} M_{i,j} b(v_j + (\sum_{k \in A \setminus \{k_0\}} r_k - v_1)u_j + a u_j)}{bz'_{k_0,\rho(i)}}} \\ &= g^{\frac{\lambda_{k_0,i}}{z_{k_0,\rho(i)}}} \end{aligned}$$

Remembering that in this case $z_{k_0,\rho(i)} := bz'_{k_0,\rho(i)}$. Finally for the queries to the other authorities $k \in A \setminus \{k_0\}$, the simulator chooses uniformly at random a vector $\vec{t_k} \in \mathbb{Z}_p^n$ such that $t_{k,1} = -r_k$ and implicitly sets the shares $\lambda_{k,i} = b\sum_{j=1}^{n} M_{i,j} t_{k,j}$ by computing

$$K_{k,i} = \begin{cases} (g^b)^{\frac{\sum_{j=1}^{n} M_{i,j} t_{k,j}}{z'_{k,\rho(i)}}} = g^{\frac{b\sum_{j=1}^{n} M_{i,j} t_{k,j}}{z'_{k,\rho(i)}}} = g^{\frac{\lambda_{k,i}}{z_{k,\rho(i)}}} & \text{if } i \in S \\ g^{\frac{\sum_{j=1}^{n} M_{i,j} t_{k,j}}{z'_{k,\rho(i)}}} = g^{\frac{b\sum_{j=1}^{n} M_{i,j} t_{k,j}}{bz'_{k,\rho(i)}}} = g^{\frac{\lambda_{k,i}}{z_{k,\rho(i)}}} & \text{if } i \notin S \end{cases}$$

Challenge The adversary gives two messages m_0, m_1 to the simulator. It flips a coin μ. It creates:

$$\begin{aligned} C' &= m_\mu \cdot Z \stackrel{*}{=} m_\mu \cdot e(g,g)^{abs} \\ &= m_\mu \cdot \left(e(g,g)^{(ab + b(\sum_{k \in A \setminus \{k_0\}} r_k)} \prod_{k \in A \setminus \{k_0\}} e(g,g)^{br_k} \right)^s \\ C_{k,i} &= (g^s)^{z'_{k,\rho(i)}} = g^{s z_{k,\rho(i)}} \qquad k \in A, \quad i \in S \end{aligned}$$

Where the equality $\stackrel{*}{=}$ holds if and only if the BDH challenge was a valid tuple (i.e. Z is non-random).

Phase II During this phase the simulator acts exactly as in *Phase I*.

Guess The adversary will eventually output a guess μ' of μ. The simulator then outputs 0 to guess that $Z = e(g,g)^{abs}$ if $\mu' = \mu$; otherwise, it outputs 1 to indicate that it believes that Z is a random group element in $\mathbb{G}_2$. In fact when Z is not random the simulator $\mathcal{B}$ gives a perfect simulation so it holds:

$$Pr\left[\mathcal{B}\left(\vec{y}, Z = e(g,g)^{abs}\right) = 0\right] = \frac{1}{2} + \epsilon$$

On the contrary when Z is a random element $R \in \mathbb{G}_2$ the message m_μ is completely hidden from the adversary point of view, so:

$$Pr\left[\mathcal{B}\left(\vec{y}, Z = R\right) = 0\right] = \frac{1}{2}$$

Therefore, $\mathcal{B}$ can play the decisional BDH game with non-negligible advantage $\frac{\epsilon}{2}$.

□

3.3.4 Remarks

This scheme gives a solution to the problem of faith in the authority, specifically the concerns arisen by key escrow and clearance check. Key escrow is a setting in which a party (in this case the authority) may obtain access to private keys and thus it can decrypt any ciphertext. Normally the users have faith in the authority and assume that it will not abuse its powers.

The problem arises when the application does not plan a predominant role and there are trust issues selecting any third party that should manage the keys. In this situation the authority is seen as *honest but curious*, in the sense that it will provide correct keys to users (then it is not malicious) but will also try to access data beyond its competence. It is clear that as long as a single authority is the unique responsible to issuing the keys, there is no way to prevent key escrow. Thus the need for multi-authority schemes arises.

Another problem addressed is more specific of KP-ABE. In this setting the authority has to assign to each user an appropriate access structure that represents what the user can and cannot decrypt. Therefore, the authority has to be trusted not only to give correct keys and to not violate the privacy, but also to perform correct checks of the users' clearance and to assign adequate access structures accordingly.

Therefore, in addition to satisfying the requirements of not being *malicious* and not being *curious*, the authority must also not have been *breached*, in the sense that a user's keys must embed access structures that faithfully represent said user's level of clearance, and that no one has access to keys with a higher level of clearance than the one they are due.

In this case, to add multiple authorities to the scheme gives to the encryptor the opportunity to request more guarantees about the legitimacy of the decryptor's clearance. In fact, each authority checks the users independently, so the idea is to request that the decryption proceeds successfully only when a key for each authority of a given set A is used. This means that the identity of the user has been checked by every selected authority, and their choice models the trust that the encryptor has in them. Note that if these authorities set up their parameters independently and during encryption these parameters are bound together irrevocably, then no authority can single-handedly decrypt any ciphertext and thus key escrow is removed. So this MA-KP-ABE schemes guarantee a protection against both breaches and curiosity.

This scheme has very short single-keys (just one element per row of the access matrix) that compensates for the need of multiple single-keys (one for cited authority) in the decryption. Ciphertexts are also quite short (the number of elements is linear in the number of authorities times the number of attributes under which it has been encrypted) thus the scheme is efficient under this aspect. Moreover, there are *no* pairing computations involved during encryption and this means significant advantages in terms of encryption times. Decryption time is not constant in the number of pairings (e.g. as in the scheme presented in [41] or the one in [85]) but requires $\sum_{k \in A} l_k$ pairings where A is the set of authorities involved in encryption and l_k is the number of rows of the access matrix of the key given by authority k, so to maintain the efficiency of the scheme only a few authorities should be requested by the encryptor.

Related Work

Taking a more historical perspective, the problem of multi-authority ABE is not novel and a few solution have been proposed. The problem of building ABE systems with multiple authorities was proposed by Sahai and Waters. This problem with the presence of a central authority was firstly considered by Chase [26] and then improved by Chase and Chow [27], constructing simple-threshold schemes in the case where attributes are divided in disjoint sets, each controlled by a different authority. These schemes are also shown to be extensible from simple threshold to KP-ABE, but retaining the partition of attributes and requiring the involvement of every authority in the decryption. In those works the main goal is to relieve the central authority of the burden of generating key material for every user and add resiliency to the system. Multiple authorities manage the attributes, so that each has less work and the whole system does not get stuck if one is down. Another approach has been made by Lin et al. [52] where a central authority is not needed but a parameter directly sets the efficiency and number of users of the scheme.

More interesting results have been achieved for CP schemes, in which the partition of the attributes makes more sense, for example [64]. The most recent and interesting result may be found in [50], where Lewko and Waters propose a scheme where there is no need for a central authority or coordination between the authorities, each controlling disjoint sets of attributes. They used composite bilinear groups and via Dual System Encryption (introduced by Waters [84] with techniques developed with Lewko [49]) proved their scheme fully secure following the example of Lewko et al. [48]. They allow the adversary to statically corrupt authorities choosing also their master key. Note however that they did not specifically address key escrow but only distributed workload.

To remark the difference of the scheme presented in this section, note that a different setting is addressed. For example a situation that suits the scheme proposed here, but not the one of Lewko and Waters is the following. Consider a company that has branches dislocated on various parts of the world, each checking its personnel and giving to each one an access policy (thus acting as authorities). This scheme allows encryptions that may be decrypted by the manager of the branch (simply use only one authority as in a classic ABE scheme) but also more secure encryptions that require the identity of the decryptor to be certified by more centres, basing the requirements on which branches are still secure and/or where a user may actually authenticate itself.

Moreover, note also that although the scheme of [50] is proven fully secure (against selective security), the construction is made in composite bilinear groups. It is in fact compulsory when using Dual System encryption, but this has drawbacks in terms of group size (integer factorization has to be avoided) and the computations of pairings and group operations are much less efficient. This fact leads to an alternative construction in prime order groups in the same paper, that however is proven secure only in the weaker generic group and random oracle model.

Implementation

A proof-of-concept of this protocol has been implemented in C using the PBC library [58] that provides low-level routines for pairing-based cryptosystems,

built on top of the GMP library [1] for arbitrary precision arithmetic over integers. The implementation has not been adequately tested in order to properly assess its performance, however no obvious problem has been noticed. This proof-of-concept demonstrates the practical viability of the proposed protocol, however for a *production-ready* solution particular care should be used regarding the heavy usage of random-generated parameters. In fact randomness is always a delicate issue and proper sources should be employed. Moreover the choice of the elliptic curve and bilinear group has heavy repercussions on both the security and the efficiency of the protocol, so a rigorous analysis should be performed in order to properly address this problem. Another problem to be considered is the size of the attribute sets and access policy since the resource usage scales quadratically in these dimensions.

3.4 Collaborative Multi-Authority Key-Policy Attribute-Based Encryption

This section is divided in three parts. First Collaborative Multi-Authority Key-Policy ABE and its CPA selective security are defined. In the second part the scheme is presented in detail and, finally, a variant of the BDH assumption (Definition 2.3) is used to prove the security of this scheme in the selective set model.

3.4.1 Collaborative Multi Authority KP-ABE Structure and Security

In this scheme, the authorities set up independently their master keys and they collaborate to create a common public key and some authority parameters that will be used to generate secret keys. There is a minimum collaboration during key generation, in the sense that authorities have to agree on the access policy to assign to the user, or equivalently the user should ask for the same policy to every authority. Note however that it is very reasonable that the same access policy is assigned since it is strictly related to the specific user. Moreover note that even if the policy of a user might contain sensitive data, it might be safely shared between authorities since they are entitled to access this kind of information anyway, and there is no randomness shared between authorities in doing so.

To encrypt, a user chooses a set of attributes that describes the message (and thus determines which access structures give access to it). The ciphertext is computed using the public key generated by the authorities in concert. When someone wants to decrypt, they need a key for every authority and once they obtains all the pieces they can merge and use them as a single key.
The formal definition of the scheme follows.

Let $\mathbb{G}_1$ be a bilinear group (chosen accordingly to an implicit security parameter λ), $g \in \mathbb{G}_1$ a generator of the group, and $\mathbb{A}$ an access structure on a universe of attributes U.

Definition 3.5 (Collaborative Multi-authority KP-ABE)**.** A collaborative multi-authority Key-Policy ABE system for a message space $\mathcal{M}$, a universe of au-

thorities X, and an access structure space $\mathcal{G}$ is composed of the following four algorithms:

$\mathsf{Setup}(U, g, \mathbb{G}_1) \to (\mathsf{PK}_k, \mathsf{MK}_k, \mathsf{AP}_k)$. The setup algorithm for the authority $k \in X$ takes as input the universe of attributes U and the bilinear group $\mathbb{G}_1$ alongside its generator g. It outputs the public parameters PK_k, the master key MK_k, and the authority parameters AP_k for that authority.

$\mathsf{CollSetup}(\mathsf{MK}_k, \mathsf{PK}_k, \mathsf{AP}_k, \mathsf{PK}^{(h)}, \mathsf{AP}^{(h)}) \to (\mathsf{PK}^{(h+1)}, \mathsf{AP}^{(h+1)})$. The collaborative part of setup asks the authority $k \in X$ to add their part to the final public key and authority parameters. It takes as input the master key MK_k for that authority and the h-th step of construction of the public key $\mathsf{PK}^{(h)}$, and of the authority parameters $\mathsf{AP}^{(h)}$. It outputs the next step of construction of the public key $\mathsf{PK}^{(h+1)}$ and authority parameters $\mathsf{AP}^{(h+1)}$ (at the first step, i.e. $h = 0$, they are simply initialized with the parameters of the first authority). When $h = x = |X|$ then $\mathsf{PK}^{(x)} = \mathsf{PK}$ and $\mathsf{AP}^{(x)} = \mathsf{AP}$ i.e. the public and authority parameters are completed once every authority has contributed. At this point PK is distributed among all users, while AP is shared only between authorities.

$\mathsf{KeyGen}_k(\mathsf{MK}_k, \mathsf{AP}, (M, \rho)) \to \mathsf{SK}_k$. The key generation algorithm for the authority $k \in X$ takes as input the master key MK_k of the authority and an access structure $\mathbb{A}$ in the form of an LSSS (M, ρ). It outputs a decryption key SK_k for that access structure.

$\mathsf{Encrypt}(m, S, \mathsf{PK}) \to \mathsf{CT}$. The encryption algorithm takes as input the public parameters PK, a message $m \in \mathcal{M}$ and a set of attributes $S \subseteq U$. It outputs the ciphertext CT associated with the attribute set S.

$\mathsf{Decrypt}(\mathsf{CT}, \{\mathsf{SK}_k\}_{k \in X}) \to m'$. The decryption algorithm takes as input a ciphertext CT that was encrypted under a set S of attributes and a decryption key SK_k for every authority $k \in X$. Let $\mathbb{A}$ be the access structure of each key SK_k. It outputs the message m' if and only if $S \in \mathbb{A}$.

The security game is defined as follows.

Definition 3.6 (CMA-KP-ABE Security Game)**.** Take a CMA-KP-ABE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt})$ for a message space $\mathcal{M}$, a universe of authorities X and an access structure space $\mathcal{G}$ and consider the following CMA-KP-ABE experiment $\mathsf{CMA\text{-}KP\text{-}ABE\text{-}Exp}_{\mathcal{A},\mathcal{E}}(\lambda, U)$ for an adversary $\mathcal{A}$, security parameter λ and attribute universe U:

Init. The adversary declares the set of attributes S that it wishes to be challenged upon. Moreover it selects the *honest authority* $k_0 \in X$.

Setup. The challenger runs the Setup and Collaborative Setup algorithms initializing the authorities, and gives to the adversary the individual public key and the authority parameters of every authority, alongside all the master keys of the non-honest authority and every collaboration step.

Phase I. The adversary issues queries for private keys generated by k_0, however the access structures $\mathbb{A}$ relative to these keys can not authorize the target set, that is $S \notin \mathbb{A}$.

Challenge. The adversary submits two equal length messages m_0 and m_1. The challenger flips a random coin $b \in \{0,1\}$, and encrypts m_b with S. The ciphertext is passed to the adversary.

Phase II. Phase `I` is repeated.

Guess. The adversary outputs a guess b' of b.

The output of the experiment is 1 if $b' = b$, 0 otherwise.

Definition 3.7 (CMA-KP-ABE Selective Security)**.** The CMA-KP-ABE scheme $\mathcal{E}$ is CPA selective secure (or secure against chosen-plaintext attacks) for attribute universe U if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function $negl$ such that:

$$\Pr[\mathsf{MA\text{-}KP\text{-}ABE\text{-}Exp}_{\mathcal{A},\mathcal{E}}(\lambda, U) = 1] \leq \frac{1}{2} + negl(\lambda).$$

3.4.2 The Scheme

This scheme plans a set X of authorities, each with their own parameters, that collaborate to create a common public key and it sets up an encryption algorithm that uses this public key so that an authorized key for each authority in X is required to successfully decrypt.

The scheme consists of three randomized algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}$) plus the collaborative step $\mathsf{CollSetup}$ and decryption $\mathsf{Decrypt}$. The scheme works in a bilinear group $\mathbb{G}_1$ of prime order p, and uses LSSS matrices to share secrets according to the various access structures. Attributes are seen as elements of $\mathbb{Z}_p$.

The description of the algorithms follows.

$\mathsf{Setup}(U, g, \mathbb{G}_1) \rightarrow (\mathsf{PK}_k, \mathsf{MK}_k, \mathsf{AP}_k)$. Given the universe of attributes U and a generator g of $\mathbb{G}_1$ each authority sets up independently its parameters. For $k \in X$ the Authority k chooses uniformly at random $\alpha_k \in \mathbb{Z}_p$, and $z_{k,i} \in \mathbb{Z}_p$ for each $i \in U$. Then the public parameters PK_k, the master key MK_k, and the authority parameters AP_k are:

$$\mathsf{PK}_k = \left(Y_k = e(g,g)^{\alpha_k}, \{T_{k,i} = g^{z_{k,i}}\}_{i \in U}\right) \tag{3.9}$$

$$\mathsf{MK}_k = (\alpha_k, \{z_{k,i}\}_{i \in U}) \tag{3.10}$$

$$\mathsf{AP}_k = \left(\left\{V_{k,i} = g^{\frac{1}{z_{k,i}}}\right\}_{i \in U}\right) \tag{3.11}$$

$\mathsf{CollSetup}(\mathsf{MK}_k, \mathsf{PK}_k, \mathsf{AP}_k, \mathsf{PK}^{(h)}, \mathsf{AP}^{(h)}) \rightarrow (\mathsf{PK}^{(h+1)}, \mathsf{AP}^{(h+1)})$. The collaborative construction of the public key proceeds as follows:

- if $h = 0$ then the authority k is the first to participate, then it simply sets $\mathsf{PK}^{(1)} = \mathsf{PK}_k$, $\mathsf{AP}^{(1)} = \mathsf{AP}_k$
- if $h > 0$ then $\mathsf{PK}^{(h)} = \left(Y^{(h)}, \left\{T_i^{(h)}\right\}_{i \in U}\right)$, $\mathsf{AP}^{(h)} = \left(\left\{V_i^{(h)}\right\}_{i \in U}\right)$, so it sets

$$Y^{(h+1)} = Y^{(h)}.Y_k, \quad T_i^{(h+1)} = \left(T_i^{(h)}\right)^{z_{k,i}}, \quad V_i^{(h+1)} = \left(V_i^{(h)}\right)^{\frac{1}{z_{k,i}}} \quad \forall i \in U$$

Then it is easy to see that when the construction is complete (i.e. every authority has contributed) the public key is:

$$\mathsf{PK}^{(x)} = \mathsf{PK} = \left(Y = e(g,g)^{\sum_{k\in X} \alpha_k}, \left\{ T_i = g^{\prod_{k\in X} z_{k,i}} \right\}_{i\in U} \right) \tag{3.12}$$

$$\mathsf{AP}^{(x)} = \mathsf{AP} = \left(\left\{ V_i = g^{\frac{1}{\prod_{k\in X} z_{k,i}}} \right\}_{i\in U} \right) \tag{3.13}$$

$\mathsf{KeyGen}_k(\mathsf{MK}_k, \mathsf{AP}, (M,\rho)) \to \mathsf{SK}_k$. The key generation algorithm for the authority k takes as input the master key MK_k, the public key PK and an LSSS access structure (M,ρ), where M is an $l \times n$ matrix on $\mathbb{Z}_p$ and ρ is a function which associates rows of M to attributes. It chooses uniformly at random a vector $\vec{v}_k \in \mathbb{Z}_p^n$ such that $v_{k,1} = \alpha_k$. Then computes the shares $\lambda_{k,i} = M_{k,i}\vec{v}_k$ for $1 \le i \le l$ where $M_{k,i}$ is the i-th row of M_k. Then the private key SK_k is:

$$\mathsf{SK}_k = \left\{ K_{k,i} = V_{\rho(i)}^{\lambda_{k,i}} = g^{\frac{\lambda_{k,i}}{\prod_{k\in X} z_{k,\rho(i)}}} \right\}_{1\le i\le l} \tag{3.14}$$

$\mathsf{Encrypt}(m, S, \mathsf{PK}) \to \mathsf{CT}$. The encryption algorithm takes as input the public key, a set S of attributes and a message m to encrypt. It chooses $s \in \mathbb{Z}_p$ uniformly at random and then computes the ciphertext as:

$$\mathsf{CT} = (S, C' = m \cdot (Y)^s, \{C_i = (T_i)^s\}_{i\in S}) \tag{3.15}$$

$\mathsf{Decrypt}(\mathsf{CT}, \{\mathsf{SK}_k\}_{k\in X}) \to m'$. The input is a ciphertext for a set of attributes S and an authorized key for every authority. Let (M,ρ) be the LSSS associated to the keys, and suppose that S is authorized. The algorithm finds $w_i \in \mathbb{Z}_p, i \in I$ such that

$$\sum_{i\in I} \lambda_{k,i} w_i = \alpha_k \qquad \forall k \in X \tag{3.16}$$

for an appropriate subset $I \subseteq S$. To simplify the notation let $z_i := \prod_{k\in X} z_{k,i}$, the algorithm then proceeds to reconstruct the original message computing:

$$\begin{aligned} m' &= \frac{C'}{\prod_{i\in I} e(\prod_{k\in X} K_{k,i}, C_{\rho(i)})^{w_i}} \\ &= \frac{m \cdot \left(e(g,g)^{(\sum_{k\in X} \alpha_k)}\right)^s}{\prod_{i\in I} e\left(\prod_{k\in X} g^{\frac{\lambda_{k,i}}{z_{\rho(i)}}}, (g^{z_{\rho(i)}})^s\right)^{w_i}} \\ &= \frac{m \cdot e(g,g)^{s(\sum_{k\in X} \alpha_k)}}{e(g,g)^{s \sum_{k\in X} \sum_{i\in I} w_i \lambda_{k,i}}} \\ &\overset{*}{=} \frac{m \cdot e(g,g)^{s(\sum_{k\in X} \alpha_k)}}{e(g,g)^{s(\sum_{k\in X} \alpha_k)}} = m \end{aligned}$$

Where $\overset{*}{=}$ follows from the property (3.16).

Note that once the user has obtained the keys from every authority they can multiply these all together and store only $\mathsf{SK} = \left\{ K_i = \prod_{k\in X} K_{k,i} \right\}_{1\le i\le l}$ since this is all they need to perform the decryption, so actually only a key of size l is needed, hence the scheme is very efficient in terms of key-size.

3.4.3 Security

The scheme is proved secure under the ABDH assumption in the selective set security game described in Definition 3.6. Recall that every authority but one is supposed curious (or corrupted or breached) and then the attacker has access to their master keys and so is able to issue even keys that have enough clearance for the target set of attributes, while the honest authority issues only unauthorized keys. Thus if at least one authority remains trustworthy the scheme is secure.

The security is provided by the following theorem.

Theorem 3.2. *If an adversary can break the scheme, then a simulator can be constructed to play the Decisional ABDH game with a non-negligible advantage.*

Proof. Suppose there exists a polynomial-time adversary $\mathcal{A}$, that can attack the scheme in the Selective-Set model with advantage ϵ. Then we claim that a simulator $\mathcal{B}$ can be built that can play the Decisional ABDH game with advantage $\epsilon/2$. The simulation proceeds as follows.

Init The simulator takes in a ABDH challenge $\vec{y} = (g, g^a, g^b, g^{\frac{1}{b}}, g^s), Z$. The adversary gives to the simulator the challenged set of attributes S, and chooses the honest authority $k_0 \in X$, where X is the set of authorities.

Setup The simulator chooses random $r_k \in \mathbb{Z}_p$ for $k \in X \setminus \{k_0\}$, sets $\alpha_k = -r_k$ for $k \in X \setminus \{k_0\}$ and implicitly sets $\alpha_{k_0} = ab + \sum_{k \in X \setminus \{k_0\}} r_k$ by computing:

$$Y_{k_0} = e(g,g)^{\alpha_{k_0}} = e(g^a, g^b) e(g,g)^{\sum_{k \in X \setminus \{k_0\}} r_k} \tag{3.17}$$

$$Y_k = e(g,g)^{\alpha_k} = e(g,g)^{-r_k} \qquad \forall k \in X \setminus \{k_0\} \tag{3.18}$$

Then it chooses $z'_{k,i} \in \mathbb{Z}_p$ uniformly at random for each $i \in U$, $k \in X$ and sets the public parameters as:

$$T_{k_0,i} = \begin{cases} g^{z'_{k_0,i}} & \text{if } i \in S \\ (g^b)^{z'_{k_0,i}} & \text{if } i \notin S \end{cases} \tag{3.19}$$

$$T_{k,i} = g^{z'_{k,i}} \qquad \text{for } k \neq k_0 \tag{3.20}$$

And the authority parameters as:

$$V_{k_0,i} = \begin{cases} g^{\frac{1}{z'_{k_0,i}}} & \text{if } i \in S \\ (g^{\frac{1}{b}})^{\frac{1}{z'_{k_0,i}}} & \text{if } i \notin S \end{cases} \tag{3.21}$$

$$V_{k,i} = g^{\frac{1}{z'_{k,i}}} \qquad \text{for } k \neq k_0 \tag{3.22}$$

The simulator can now pass the master keys of the non-honest authorities, and every public and authority parameter to the adversary.

Then it proceeds to simulate the collaborative steps of the scheme. To formalize this let us introduce an ordering function $\psi : X \to \{j : 1 \leq j \leq |X|\}$ that simply specifies in which order the authorities collaborate (that is if $\psi(k_0) =$

1 then the honest authority begins the collaboration). Then the collaborative steps of the public parameters (for $i \in U$) are computed as:

$$Y^{(h)} = \begin{cases} e(g^a, g^b)e(g,g)^{\prod_{k\in X:\psi(k)>h} r_k} & \text{if } \psi(k_0) \leq h \\ e(g,g)^{-\prod_{k\in X:\psi(k)\leq h} r_k} & \text{otherwise} \end{cases} \tag{3.23}$$

$$T_i^{(h)} = \begin{cases} (g^b)^{\prod_{k\in X:\psi(k)\leq h} z'_{k,i}} & \text{if } i \notin S \wedge \psi(k_0) \leq h \\ g^{\prod_{k\in X:\psi(k)\leq h} z'_{k,i}} & \text{otherwise} \end{cases} \tag{3.24}$$

$$V_i^{(h)} = \begin{cases} (g^b)^{\frac{1}{\prod_{k\in X:\psi(k)\leq h} z'_{k,i}}} & \text{if } i \notin S \wedge \psi(k_0) \leq h \\ g^{\frac{1}{\prod_{k\in X:\psi(k)\leq h} z'_{k,i}}} & \text{otherwise} \end{cases} \tag{3.25}$$

Using the previously introduced notation $z'_i := \prod_{k\in X} z'_{k,i}$, for $h = |X|$ we have the complete public key:

$$Y = e(g,g)^{ab} \tag{3.26}$$

$$T_i = \begin{cases} g^{z'_i} & \text{if } i \in S \\ g^{bz'_i} & \text{if } i \notin S \end{cases} \tag{3.27}$$

$$V_i = \begin{cases} g^{\frac{1}{z'_i}} & \text{if } i \in S \\ g^{\frac{1}{bz'_i}} & \text{if } i \notin S \end{cases} \tag{3.28}$$

Phase I In this phase the simulator answers to private key queries made to the honest authority k_0. The simulator has to compute the $K_{k_0,i}$ values of a key for an access structure (M, ρ) with dimension $l \times n$ that is not satisfied by S. Therefore for the property 3.2 of an LSSS it can find a vector $\vec{u} \in \mathbb{Z}_p^n$ with $u_1 = 1$ such that

$$M_i \vec{u} = 0 \qquad \forall i \text{ such that } \rho(i) \in S \tag{3.29}$$

Then it chooses uniformly at random a vector $\vec{v} \in \mathbb{Z}_p^n$ and implicitly sets the shares of $\alpha_{k_0} = ab + \sum_{k\in X\setminus\{k_0\}} r_k$ as

$$\lambda_{k_0,i} = \sum_{j=1}^{n} M_{i,j}(bv_j + (ab + \sum_{k\in X\setminus\{k_0\}} r_k - bv_1)u_j) \tag{3.30}$$

Note that $\lambda_{k_0,i} = \sum_{j=1}^{n} M_{i,j} w_j$ where $w_j = bv_j + (ab + \sum_{k\in X\setminus\{k_0\}} r_k - bv_1)u_j$ thus $w_1 = bv_1 + (ab + \sum_{k\in X\setminus\{k_0\}} r_k - bv_1)1 = ab + \sum_{k\in X\setminus\{k_0\}} r_k = \alpha_{k_0}$ so the shares are valid. Note also that from (3.29) it follows that

$$\begin{aligned} \lambda_{k_0,i} &= \sum_{j=1}^{n} M_{i,j} b v_j + \sum_{j=1}^{n} M_{i,j} u_j (ab + \sum_{k\in X\setminus\{k_0\}} r_k - bv_1) \\ &= b \sum_{j=1}^{n} M_{i,j} v_j \quad \forall i \text{ such that } \rho(i) \in S \end{aligned}$$

Thus if i is such that $\rho(i) \in S$ the simulator can compute

$$K_{k_0,i} = (g^b)^{\frac{\sum_{j=1}^{n} M_{i,j} v_j}{z'_{\rho(i)}}} = g^{\frac{\lambda_{k_0,i}}{z_{\rho(i)}}}$$

Otherwise, if i is such that $\rho(i) \notin S$ the simulator computes

$$\begin{aligned} K_{k_0,i} &= g^{\frac{\sum_{j=1}^{n} M_{i,j}(v_j - v_1 u_j)}{z'_{\rho(i)}}} (g^a)^{\frac{\sum_{j=1}^{n} M_{i,j} u_j}{z'_{\rho(i)}}} (g^{\frac{1}{b}})^{\frac{\sum_{j=1}^{n} M_{i,j} u_j \sum_{k \in X \setminus \{k_0\}} r_k}{z'_{\rho(i)}}} \\ &= g^{\frac{b \sum_{j=1}^{n} M_{i,j}(v_j - v_1 u_j)}{b z'_{\rho(i)}}} g^{\frac{ab \sum_{j=1}^{n} M_{i,j} u_j}{b z'_{\rho(i)}}} g^{\frac{\sum_{j=1}^{n} M_{i,j} u_j \sum_{k \in X \setminus \{k_0\}} r_k}{b z'_{\rho(i)}}} \\ &= g^{\frac{\sum_{j=1}^{n} M_{i,j}(b v_j + (ab + \sum_{k \in X \setminus \{k_0\}} r_k - b v_1) u_j)}{b z'_{\rho(i)}}} \\ &= g^{\frac{\lambda_{k_0,i}}{z_{\rho(i)}}} \end{aligned}$$

Where the last equality follows from $z_{\rho(i)} = b z'_{\rho(i)}$.

Note that the adversary has the master keys of the other authorities, so they can create any other private key.

Challenge The adversary gives two messages m_0, m_1 to the simulator, that flips a coin μ and creates:

$$\begin{aligned} C' &= m_\mu \cdot Z \stackrel{*}{=} m_\mu \cdot e(g,g)^{abs} = m_\mu Y^s \\ C_{k,i} &= (g^s)^{z'_{\rho(i)}} = g^{s z_{\rho(i)}} \qquad i \in S \end{aligned}$$

Where the equality $\stackrel{*}{=}$ holds if and only if the ABDH challenge was a valid tuple (i.e. Z is non-random).

Phase II During this phase the simulator acts exactly as in *Phase I*.

Guess The adversary will eventually output a guess μ' of μ. The simulator then outputs 0 to guess that $Z = e(g,g)^{abs}$ if $\mu' = \mu$; otherwise, they outputs 1 to indicate that they believes Z is a random group element in $\mathbb{G}_2$. In fact when Z is not random the simulator $\mathcal{B}$ gives a perfect simulation so it holds:

$$Pr\left[\mathcal{B}\left(\vec{y}, Z = e(g,g)^{abs}\right) = 0\right] = \frac{1}{2} + \epsilon$$

On the contrary when Z is a random element $R \in \mathbb{G}_2$ the message m_μ is completely hidden from the adversary point of view, so:

$$Pr\left[\mathcal{B}\left(\vec{y}, Z = R\right) = 0\right] = \frac{1}{2}$$

Therefore, $\mathcal{B}$ can play the decisional BDH game with non-negligible advantage $\frac{\epsilon}{2}$. □

3.4.4 Remarks

This construction evolves from the previous scheme exploiting the collaboration between authorities to improve the efficiency. This scheme needs fewer parameters, since the collaboration permits to collapse the various public parameters in a single public key, significantly reducing the length of ciphertexts. Moreover, once all the single-keys have been obtained they may be collapsed into one:

$$SK = \left\{ K_i = \prod_{k \in X} K_{k,i} = g^{\frac{\sum_{k \in X} \lambda_{\bar{k},i}}{\prod_{k \in X} z_{k,\rho(i)}}} \right\}_{1 \le i \le l}. \tag{3.31}$$

This scheme requires that each authority uses the same LSSS matrix to generate the single-key, but the assumption is not unreasonable since the matrix is directly derived from the user's clearance. So for the price of the collaboration steps that weigh down the setup (a phase that has to be executed only once when the scheme is used), and an additional parameter shared by authorities, we obtain great improvement in the encryption, decryption and key-storage.

Chapter 4

Format Preserving Tokenization Algorithm for Credit Cards

Credit cards are widely used in online shopping, and given the large amount of threats that permeate the web their security is a major concern for millions of people around the world. Tokenization allows users to make payments without exposing the credit card directly, thus limiting the risks.

In this chapter an original tokenization algorithm is presented, that preserves the format of the PAN and complies with the PCI standard, and its security is proven.

This original work has been published in the Journal *Applicable Algebra in Engineering, Communication and Computing* [5], a joint work with Dr. Riccardo Aragona and Prof. Massimiliano Sala. The graduand proved the security of the protocol, while the coauthors designed it and collaborated to translate the proof to compliance with the industry standard.

4.1 Introduction

In recent years, credit cards have become one of the most popular payment instruments. Their growing popularity has brought many companies to store the card information of its customers to make simpler subsequent payments. This need is shared by many other actors in the payment process. On the other hand, credit card data are very sensitive information, and the theft of such data is considered a serious threat.

Any company that stores credit card data aims to achieve the *Payment Card Industry Security Standard Council* (*PCI SSC*) compliance. The PCI SSC is an organization, founded by the largest payment card networks, which has developed several standards and recommendations. One of these is called the *PCI Data Security Standard* (*PCI DSS* [79]) and its goal is to guarantee the security of credit card data. PCI DSS requires that companies that handle payment cards protect the data of the cardholder when these are stored, transmitted or

processed.

These stringent requirements led to consider a new method for storage and transmission of the card information: instead of protecting the actual card data, it is easier to remove them (when their storing is not requested) and replace them with another value, called *token*. Tokens are alpha-numeric strings representing the *PAN* (*Primary Account Number*) of a payment card, and may have a similar format. In any case, from a token it must be infeasible (without additional information) to recover the PAN from which it was generated. This process is called *tokenization*. In [31], the authors present an interesting formal cryptographic study of tokenization systems and their security.

In recent years, PCI SSC has drafted some guidelines to design a tokenization solution [69, 78]. In [78], the following five types of tokens are described: *Authenticatable Irreversible Tokens*, *Non-Authenticatable Irreversible Tokens*, *Reversible Cryptographic Tokens*, *Reversible Non-Cryptographic Tokens* and *Reversible Hybrid Tokens*.
In particular, a reversible tokenization algorithm , i.e., providing the possibility for entities using or producing tokens to obtain the original PAN from the token, can be designed in three ways:

- *Reversible Cryptographic*, if it generates tokens from PANs using strong cryptography. In particular, a mathematical relationship between PAN and corresponding token exists. In this case, the PAN is never stored; only the cryptographic key is (securely) stored.
- *Reversible Non-Cryptographic*, if obtaining a PAN from its token is only by a data look-up in a dedicated server (called a Card Data Vault). In this case, the token has no mathematical relationship with its associated PAN and the only thing to be kept secret is the actual relationship between the PAN and its token (e.g., a look-up table in the Card Data Vault).
- A reversible tokenization algorithm is called *Hybrid* if it contains some features of both Reversible Cryptographic tokens and Reversible Non-Cryptographic tokens. A typical situation of this type is when, although there is a mathematical relationship between a token and its associated PAN, a data look-up table must be used to retrieve the PAN from the token.

The tokenization algorithm presented in this chapter is of the *Reversible Hybrid* type, based on a block cipher with a secret key and (possibly public) additional input. To fully appreciate the design and proposed proofs it is necessary to analyse the PCI requirements in more detail.

The organization of this chapter is the following:

- In Section 4.2 some PCI requirements for a tokenization algorithm ([69, 78]) are analysed.
- In Section 4.3 the tokenization algorithm is described in detail;
- In Section 4.4 the security of the algorithm defined in Section 4.3 is proven in a very general scenario, which would imply that the algorithm satisfies most requirements present in Section 4.2. More precisely, the tokenization algorithm is proven IND-CPA secure.

- In Section 4.5, an instantiation of the algorithm is given, considering concrete cryptographic primitives and fixing PAN length, and security and efficiency are analysed in this real-life application.

4.2 Background: Requirements of the Standard

In every industry the presence of a standard allows multiple parties to create devices compatible one with another, and security standards guarantee that some general measures are adopted and makes it easier to evaluate the safety of products and services. Thus every innovation that seeks direct applications in any industry should comply to the standard guidelines.

For this very reason the main requirements of PCI DSS for tokenization algorithms are presented here. Note that the first two requirements are not linked to security:

- Although tokens can enjoy a variety of formats, the most convenient is probably the same format of the PAN itself, since in this case a token can move inside a payment network and also be used as a payment token (for a definition of a *payment token* see p. 13 in [34]). But trying to create a token by a direct encryption of the PAN there are several problems, since the output of the encryption may not have the correct format. Besides, keep in mind that PCI requests the use of standard encryption algorithms, and this rules out the creation of ad-hoc cryptographic primitives. So the problem becomes designing algorithms that preserve the message format, the so-called *Format Preserving Encryption* (*FPE*) [14]. In literature, there are some interesting examples of algorithms that solve such problem [15, 24, 40, 62, 80].

- It must be possible to obtain different tokens from a single PAN (even one per transaction, if necessary), so the tokenization algorithm will require additional inputs, such as, a transaction counter, an expiration date, etc.

Concerning security issues, there are many requirements that an algorithm has to satisfy in order to meet PCI compliance. The main requests are:

A1 " *the recovery of the original PAN must not be computationally feasible knowing only the token or a number of tokens.*" (p. 6 in [69]).
In other words, even if an attacker has managed to collect many tokens, all coming from the same PAN, possibly even on a long period of time, they must be computationally unable to retrieve the corresponding PAN. This is a form of ciphertext-only attack.

A2 "*access to multiple token-to-PAN pairs should not allow the ability to predict or determine other PAN values from knowledge of only tokens.*" (p. 6 in [69] and GT4 in [78]).
This is a known-plaintext attack.

A3 "*Tokens should have no value if compromised or stolen, and should be unusable to an attacker if a system storing only tokens is compromised*" (p. 6 in [69]).
Since this sentence comes immediately after A1 and A2, which aim at

preventing PAN recoveries, it is reasonable to take the goal of this rather cryptic sentence to be the prevention of unauthorized token generation. In other words, an attacker possessing many tokens (but not knowing the corresponding PANs) must be unable to generate even *one* other valid token. This condition is drastically different from A1, because here it is not required for the attacker to be able to deduce any of the involved PANs.

A4 "*Converting from a token produced under one cryptographic key to a token produced under another cryptographic key should require an intermediate PAN state—i.e., invocation of de-tokenization.*" (GT 11 in [78]).
Since the system presented here uses a block cipher with additional (public) input, the interpretation could be that for an attacker that gets a token obtained by a PAN, a secret key and an additional input, it must be infeasible to compute any token corresponding to the same PAN (and same additional input) but to a different key.

A* "*The recovery of the original PAN should be computationally infeasible knowing only the token, a number of tokens, or a number of PAN/token pairs*" (GT 5 in [78]).
This is a repetition of A1 and A2.

To prove that this tokenization algorithm satisfies A1, A2 and A3,it will be proven in Section 4.4 that it satisfies an even stronger condition, under the assumption of the strength of the core primitive used to define it (a block cipher). Requirement A4 requires a separate proof in Section 4.4.

4.3 Algorithm

A card number is formed by three concatenated parts: the IIN (also called BIN), that identifies the card Issuer, a numeric code, that identifies the account, and a check digit. It is possible to assume to replace the IIN with another fixed code (called a "token BIN" in [34], p. 14), which marks the resulting card number as a token, so the first segment will be ignored in the description of the algorithm. Also the check digit can be computed normally, so it can be discarded too in the forthcoming formal description of the algorithm.

Assume that it is possible to invoke the encryption function of a block cipher E, just by sending a plaintext and obtaining a ciphertext, with a key that is kept somewhere protected and that its knowledge is not needed. Let $\mathbb{K}$ denote the keyspace of E and let $K \in \mathbb{K}$ be a key, so that E can be seen as a function $E : \mathbb{K} \times (\mathbb{F}_2)^m \longrightarrow (\mathbb{F}_2)^m$ for some $m \in \mathbb{N}$. With K fixed, E is a permutation acting on the set $(\mathbb{F}_2)^m$ of the m-bit strings. With standard block ciphers $m = 64$ or $m = 128$. Assume as usual that the set of its encryption functions forms a random sample of the set of permutations acting on $(\mathbb{F}_2)^m$.
For strings an example of the notation used is $|0110|_2$, where the index 2 denotes that only symbols from $\{0, 1\}$ are used, i.e. remainders of division by 2.

The tokenization algorithm processes two inputs: a numeric code coming from the PAN and an additional input.

- The caption *numeric code* will denote a string of ℓ decimal digits, with $\ell \in \mathbb{N}$ any agreed number, and the set of numeric codes is formally defined

as $\mathbb{P} := \{0, 1, \ldots, 9\}^\ell$. Nowadays $13 \leq \ell \leq 19$ for numeric codes coming from PANs [34], but no limitation is needed. Note that $\mathbb{P}$ is in obvious bijection with the integer set $\{a \in \mathbb{N} \mid 0 \leq a < 10^\ell\}$, and so a numeric code can also be seen as a non-negative integer, but care has to be taken to pass from one representation to the other. Let $[y]_b^s$ denote the representation (string) of a positive integer $y < b^s$ in base b with s digits, where the most significant digits are on the *left*.
For example, $[12]_{10}^2 = |12|_{10}$, $[12]_{10}^3 = |012|_{10}$ and $[13]_2^5 = |01101|_2$.
Then any positive integer X such that $X < 10^\ell$ can be easily converted to $[X]_{10}^\ell \in \mathbb{P}$. A bar will be used to denote the conversion from a string to a number, like $\overline{|12|_{10}} = 12$ and $\overline{|01100|_2} = 12$.

- The role of the additional input is to allow the presence of different tokens corresponding to the same PAN. It can be anything, such as a transaction counter or an integer denoting the current time. Formally, it will be identified as a binary string of finite but arbitrary length (as for example the binary representation of a transaction counter). Let $\mathbb{U}$ be the set containing all these strings, so that any $u \in \mathbb{U}$ is implicitly meant to be an additional input.

Let $n := \lceil \log_2(10^\ell) \rceil$ be the maximum number of bits required to represent a number with ℓ decimal digits. Since most of the block ciphers used in real life applications have a block-size of at least 64 bits, and for the maximum length of a PAN $\ell = 19$ we have $n = 64$, it is safe to assume that ℓ is such that $n \leq m$.

The second *ingredient* of the algorithm is f, a public function

$$f : \mathbb{U} \times \mathbb{P} \longrightarrow (\mathbb{F}_2)^{m-n} \tag{4.1}$$

In other words, given an additional input $u \in \mathbb{U}$ and a numeric code $X \in \mathbb{P}$ coming from the PAN, f returns a string of $m - n$ bits.
The requirement on f is to be *collision-resistant*, that is, it must be computationally infeasible to obtain two distinct pairs (X_1, u_1) and (X_2, u_2) such that $f(u_1, X_1) = f(u_2, X_2)$. This requirement compels the image space to have a size large enough to prevent brute-force collision attacks. So in the case of PAN tokenization only block ciphers with block size of at least 128 bits can be considered, in order to have an image space of dimension at least 64 bits.

The purposes of this function f are the followings:

- to pad the input of the tokenization algorithm to match the block size of the cipher;
- to allow the creation of multiple different tokens from the same PAN using the same key, useful for example to change token for each transaction.

The output of f could be seen as a *tweak* in the context of Tweakable Encryption [53]. An example for this f could be a truncated version of a cryptographic hash function.

The third *ingredient* of the tokenization algorithm is a database stored securely somewhere that contains a look-up table of PAN-token pairs. Once that a token is generated, it is inserted in the table. However, to generate a new token, it is needed to access the database only via a function check that checks if the token is already stored and returns either True or False accordingly.

One of the goals of a tokenization algorithm is to obtain an integer with ℓ decimal digits starting from another integer with the same length. Since $n := \lceil \log_2(10^\ell) \rceil < m$, only a fraction of the output of E has to be considered, in particular the n least significant bits of the output, and then this string will be converted back to an integer. However, given that $10^l < 2^n$, this integer could have $\ell + 1$ decimal digits. To solve this problem it is employed a method known as *Cycle Walking Cipher* [20], designed to encrypt messages from a space $\mathcal{M}$ using a block cipher that acts on a space $\mathcal{M}' \supset \mathcal{M}$, and obtain ciphertexts that are in $\mathcal{M}$.

Now all the elements have been given to understand the algorithm.
The *Tokenization Algorithm* $T(K, X, u)$ executes the following steps:

1. $t := f(u, X) \mid\mid \left[\bar{X}\right]_2^n$
2. $c := E(K, t)$
3. if $(\bar{c} \mod 2^n) \geq 10^\ell$, then $t := c$ and go back to step 2
4. $\texttt{token} := [\bar{c} \mod 2^n]_{10}^\ell$
5. if check($\texttt{token}$) = True, then $u := u + 1$ and go back to step 1
6. return $\texttt{token}$

The correctness of Algorithm T is obvious, so the focus is now on termination.

At Step 3 it checks if $(\bar{c} \mod 2^n) \geq 10^\ell$. Since $x \mapsto E(K, x)$ is a random permutation, c is expected to resemble a random binary string in $(\mathbb{F}_2)^m$. Therefore, the number $(\bar{c} \mod 2^n)$ is a random integer in $\{0, \ldots, 2^n - 1\}$. Recall that $2^{n-1} < 10^\ell < 2^n$. Therefore, the condition at Step 3 is met with probability $0 < p = \frac{2^n - 10^\ell}{2^n} < 1$. Going back to step 2 another pseudo-random number is computed and the probability that the condition of Step 3 is not satisfied is again p. Since the two events can be considered independent, due to the pseudorandomness property of E, the probability of the joint event goes down to p^2, and so on. Therefore, the probability that the algorithm remains stuck at Step 3 is negligible.

At Step 5 the algorithm checks if the new token is already present in the database. If that is the case, u is increased to $u + 1$. Note that the number of possible tokens is significantly greater than the number of tokens in use, so the probability it happens is small. Moreover the algorithm remains stuck only if $f(u', X) = f(u' + 1, X)$ and the tokens generated from $(\bar{u}, X)$ are already in the database for every $u \leq \bar{u} \leq u'$, but this happens very rarely thanks to the collision resistance of f.

For a more detailed discussion of the probability to meet the conditions at Step 3 and at Step 5, see the instantiation of the algorithm given in Section 4.5.

4.4 Proof of Security

In this section the algorithm previously defined will be proven secure in an Indistinguishability under a Chosen-Plaintext Attack scenario, under the condition that its core encrypting algorithm is secure in the same scenario. This

will guarantee in particular the security requirements A1, A2 and A3 recalled in 4.2. Then A4 will be proven separately.

An IND-CPA game for a tokenization algorithm T is defined analogously to the IND-CPA game for a symmetric cypher (see 2.11), where messages are replaced by numeric codes and additional inputs, while ciphertexts are replaced by tokens. So the adversary chooses two code/additional input pairs and tries to distinguish which of these corresponds to the token returned by $\mathcal{C}$. The adversary is also able to request other tokens (corresponding to a polynomial number of pairs), that may be chosen adaptively.

Definition 4.1 (IND-CPA for a Tokenization Algorithm)**.** Let $T(K, X, u) \rightarrow \texttt{token}$ be a tokenization algorithm that takes as input a key K, a numeric code $X \in \mathbb{P}$ and an additional input $u \in \mathbb{U}$, and returns a token `token`. An *Indistiguishability under Chosen Plaintext Attack* (IND-CPA) game for T between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ proceeds as follows:

Phase I $\mathcal{A}$ chooses (X_i, u_i) and sends it to $\mathcal{C}$, that responds with $\texttt{token}_i = T(K, X_i, u_i)$. This phase is repeated a polynomial number of times.

Challenge $\mathcal{A}$ chooses $(X_0^*, u_0^*), (X_1^*, u_1^*)$ (with (X_0^*, u_0^*) and (X_1^*, u_1^*) never chosen in Phase I) and sends them to $\mathcal{C}$, that selects $\nu \in \{0, 1\}$ at random and computes $\texttt{token} = T(K, X_\nu^*, u_\nu^*)$, which sends to $\mathcal{A}$.

Phase II Phase I is repeated, with the obvious restriction that $\mathcal{A}$ cannot choose (X_0^*, u_0^*) or (X_1^*, u_1^*).

Guess $\mathcal{A}$ guesses $\nu' \in \{0, 1\}$, and wins if $\nu' = \nu$.

The advantage $Adv_{\mathcal{A}}^T$ of $\mathcal{A}$ winning the IND-CPA game for the Tokenization Algorithm T is:

$$Adv_{\mathcal{A}}^T = \left| Pr\left[\nu' = \nu\right] - \frac{1}{2} \right|$$

T is said to be secure in a IND-CPA scenario if there is no probabilistic polynomial-time algorithm $\mathcal{A}$ that wins the IND-CPA game with more than negligible advantage.

Theorem 4.1 (IND-CPA Security of Tokenization Algorithm)**.** *Let T be the Tokenization Algorithm described in Section 4.3, and let E be the block cipher used in Step 2 of T. If E is secure in an IND-CPA scenario then T is secure in an IND-CPA scenario.*

Proof. Let $\mathcal{C}$ be the challenger in the IND-CPA game for E, and $\mathcal{A}$ be an algorithm that can win the IND-CPA game for T with more than negligible advantage ϵ. Then a simulator $\mathcal{S}$ can be built that plays the IND-CPA game for E by simulating an IND-CPA game for T and interacting with $\mathcal{A}$.

In the phases I and II $\mathcal{S}$ has to answer to tokenization queries (X_i, u_i) made by $\mathcal{A}$. The function f is publicly known, so $\mathcal{S}$ may compute $t_i := f(u_i, X_i) \mid\mid \left[\overline{X_i}\right]_2^n$ and queries $\mathcal{C}$ for the encryption of t_i, obtaining c_i in response. If $(\overline{c}_i \mod 2^n) \geq 10^l$, then $\mathcal{S}$ queries again $\mathcal{C}$, this time requiring the encryption of c_i, repeating this passage until $\mathcal{C}$ answers with c_i such that $(\overline{c}_i \mod 2^n) < 10^\ell$ (which will eventually happen since $c \mapsto E(K, c)$ is a random permutation). At this point $\mathcal{S}$ answers to the query of $\mathcal{A}$ with $\texttt{token}_i := [\overline{c}_i \mod 2^n]_{10}^l$.

Observe that the c_is, and hence the $\texttt{token}_i$s, will be distinct with high probability, since f is collision-resistant, even if the X_i's are identical, as long as the pairs (X_i, u_i)'s are distinct.

In the challenge phase, $\mathcal{S}$ receives from $\mathcal{A}$ two pairs $(X_0^*, u_0^*), (X_1^*, u_1^*)$. $\mathcal{S}$ computes $t_j^* := f(u_j^*, X_j^*) \mid\mid [\overline{X^*}_j]_2^n$ for $j \in \{0, 1\}$ and submits them to $\mathcal{C}$ for the challenge phase of the IND-CPA game for E. $\mathcal{C}$ chooses at random $\nu \in \{0, 1\}$ and will respond with the challenge ciphertext c. If $(\overline{c} \mod 2^n) \geq 10^l$, $\mathcal{S}$ queries $\mathcal{C}$ requiring the encryption of c, repeating this passage until $\texttt{token} := (\overline{c} \mod 2^n) < 10^\ell$ (which will eventually happen since $c \mapsto E(K, c)$ is a random permutation). At this point $\mathcal{S}$ sends $\texttt{token}$ to $\mathcal{A}$ as the challenge token of the IND-CPA game for T.

Eventually $\mathcal{A}$ will send to $\mathcal{S}$ its guess ν' for which code has been tokenized into $\texttt{token}$. $\mathcal{S}$ then forwards this guess to $\mathcal{C}$. It is clear that $\mathcal{A}$ guesses correctly if and only if $\mathcal{S}$ guesses correctly since the simulation is seamless.

Note that during the Challenge phase $\mathcal{S}$ is not allowed to send to $\mathcal{C}$ messages submitted during Phase I, and in Phase II $\mathcal{S}$ is not allowed to send to $\mathcal{C}$ the two messages submitted in the Challenge phase. Since the same restriction applies to the interaction between $\mathcal{A}$ and $\mathcal{S}$, problems may arise only when $\mathcal{S}$ queries for the re-encryption of ciphertexts to meet the condition $(\overline{c}_i \mod 2^n) < 10^\ell$. However the number of queries is polynomial and the encryption function $c \mapsto E(K, c)$ is a random permutation, so the probability of such a collision is negligible.

Finally, throughout the simulation Step 5 of the algorithm has been ignored, always supposing that the check function outputs False (to be coherent the check function has to only check the simulated tokens already generated by $\mathcal{S}$).

In phases I and II, when $\text{check}(\texttt{token}_i) = \mathsf{True}$, $\mathcal{S}$ is able to follow the algorithm properly adjusting u_i and querying $\mathcal{C}$.

For the challenge phase instead we have to hope that $\text{check}(\texttt{token}) = \mathsf{False}$, which happens with non-negligible probability ρ_{q_1} (this probability depends on the number of queries q_1 made in Phase I). When $\text{check}(\texttt{token}) = \mathsf{True}$, we cannot simulate correctly, so $\mathcal{S}$ may directly guess randomly, and so the probability of guessing is $\frac{1}{2}$ in this case. Thus the advantage ϵ' of $\mathcal{S}$ is:

$$\epsilon' = \left| \left(\rho_{q_1} \left(\frac{1}{2} + \epsilon \right) + (1 - \rho_{q_1}) \frac{1}{2} \right) - \frac{1}{2} \right|$$

which is non-negligible since ϵ and ρ_{q_1} are non-negligible. Thus $\mathcal{S}$ has a non-negligible advantage winning the IND-CPA game for E. □

As explained in Chapter 2, an encryption algorithm which is IND-CPA secure is also secure against a *known-plaintext attack*, and even more so against a *ciphertext-only* attack. Similar results will now be shown for this tokenization algorithm by proving some of the requirements presented in Section 4.2.

Corollary 4.1 (A1-A2-A3)**.** *Let T be the Tokenization Algorithm described in Section 4.3, and let E be the block cipher used in Step 2 of T. If E is secure in an IND-CPA scenario then T satisfies A1, A2 and A3.*

Proof. For each requirement, suppose that it is not met, then it is possible to build an adversary that wins the IND-CPA game.

A1 If T does not satisfy A1 then an attacker $\mathcal{A}$ can obtain a PAN from a token with enough tokens corresponding to the same PAN. So suppose that $\mathcal{A}$

has access to an algorithm $\mathcal{B}$ that takes in input a polynomial number N of tokens and with a non-negligible probability outputs

- X, if all the N tokens correspond to X;
- False, otherwise.

$\mathcal{A}$ tries the IND-CPA game for T and chooses $N-1$ pairs (X, u_i), with the same X, and sends them to $\mathcal{C}$, who responds with $\texttt{token}_i = T(K, X, u_i)$. Then $\mathcal{A}$ passes to the Challenge Phase and chooses u_0^* and (X_1^*, u_1^*), with $X_1^* \neq X$ and sends (X_0^*, u_0^*), with $X_0^* = X$, and (X_1^*, u_1^*) to $\mathcal{C}$ that selects $\nu \in \{0, 1\}$ at random and computes $\texttt{token} = T(K, X_\nu^*, u_\nu^*)$. Then $\mathcal{C}$ sends token to $\mathcal{A}$.
Then $\mathcal{A}$ runs $\mathcal{B}$ passing as inputs $\{\texttt{token}_i\}_{1 \leq i \leq N-1}$ and token, obtaining with non-negligible probability ϵ either $X_0^* = X$ or False. So $\mathcal{B}$ is able to invert the tokenization if and only if $\nu = 0$ and so $\mathcal{A}$ can guess ν and win the IND-CPA game with the same non-negligible probability ϵ.

A2 If A2 does not hold, then $\mathcal{A}$ has access to an algorithm $\mathcal{B}$ that takes two inputs (a polynomial number N of token-to-PAN pairs and a token), and returns the PAN corresponding to the token with non-negligible probability. Thanks to the conventions in Section 4.3, the algorithm inputs can be seen to be actually N pairs of type $(X_i, \texttt{token}_i)$ (plus the single token $\texttt{token}^*$) and the algorithm output to be the $X^* \in \mathbb{P}$ corresponding to $\texttt{token}^*$.

The adversary tries the IND-CPA game for T by choosing N random (X_i, u_i) and obtaining from C their corresponding tokens $\texttt{token}_i$.
Then $\mathcal{A}$ passes to the Challenge Phase and sends two random pairs: (X_0^*, u_0^*) and (X_1^*, u_1^*). The Challenger returns the token token corresponding to one of these.
Then $\mathcal{A}$ runs $\mathcal{B}$ passing as inputs the pairs $(X_i, \texttt{token}_i)$ and token, obtaining with non-negligible probability the correct $X \in \{X_0^*, X_1^*\}$, that $\mathcal{A}$ uses to guess ν, thus winning the IND-CPA game.

A3 If A3 does not hold, then it may be assumed that $\mathcal{A}$ has access to an algorithm $\mathcal{B}$ that takes as input a polynomial number N of tokens and one additional input u^*, and that returns a pair $(X, \texttt{token}^*)$, where $\texttt{token}^*$ is the token corresponding to (X, u^*) with non-negligible probability.

The adversary tries the IND-CPA game for T by choosing N random (X_i, u_i)'s and obtaining from C their corresponding tokens $\texttt{token}_i$'s. Then $\mathcal{A}$ chooses a $u^* \in \mathbb{U}$ and passes it along with these tokens to $\mathcal{B}$, obtaining X and $\texttt{token}^*$.

Then $\mathcal{A}$ passes to the Challenge Phase and sends the pair (X, u^*) and a random pair. The Challenger returns the token token corresponding to one of these. The adversary easily wins the game just by checking if $\texttt{token} = \texttt{token}^*$.

□

Theorem 4.2 (A4 for Tokenization Algorithm)**.** *Let $K, K^* \in \mathbb{K}$, $X \in \mathbb{P}$ and $u \in \mathbb{U}$. Knowing only u and the token $\texttt{token} = T(K, X, u)$, it is possible to compute $\texttt{token}^* = T(K^*, X, u)$ only with negligible probability.*

Proof. The two tokens come directly from the encryption of the same string $M = f(u, X) \,||\, [\overline{X}]_2^n$ with two different keys, except when the unlikely condition in Step 3 is met. So suppose that, with non-negligible probability, $\mathcal{A}$ is able to compute $E(K^*, M)$ from $E(K, M)$. This means that for a large portion of the plaintext space the two encryption functions are closely correlated, since from one it is possible to deduce the other without the need for decryption/re-encryption. This contradicts the first assumption on E, that is, that the set of its encryption function forms a random sample of the set of permutations acting on $(\mathbb{F}_2)^m$. □

Remark 4.1. The requirement GT5 in [78] "*The recovery of the original PAN should be computationally infeasible knowing only the token, a number of tokens, or a number of PAN/token pairs*" directly follows from A1 and A2.

Note that throughout the proofs prefixes are supposed independent from the code to be tokenized, thus they might be selected at random from a set of valid prefixes and still obtain valid tokens. In the context of PAN tokenization, the prefix is the token BIN, that is in fact independent from the central digits of the PAN and only used to route the token to the correct card issuer for detokenization. Also in this model BINs has not been considered since they are publicly available and it is not unreasonable to derive them from token BINs and vice-versa, without altering the probability of succeeding in a sensible manner.

4.5 A practical example

Tokenization is a problem of practical interest, so to conclude the chapter an example of an instantiation with concrete cryptographic primitives and fixed length of the PAN is given, and its efficiency and security will be analysed.

Consider PANs with length $\ell = 16$, so $n = 54$, take AES-256 as the cipher E, and as the function f take SHA-256 truncated to $128 - 54 = 74$ bits.

4.5.1 Security

As explained in Section 2.2.2 SHA-256 can be considered collision-resistant and so it satisfies the purposes of the tokenization algorithm (see, for instance, Step 5 in Section 4.3). Again, as stated in Section 2.2.3, AES-256 can be considered IND-CPA secure and so it satisfies the hypothesis of 4.1 and 4.1, and the randomness requirements of 4.2.

4.5.2 Efficiency

The probability that the condition at Step 3 is met is

$$p = \frac{2^{54} - 10^{16}}{2^{54}} \approx 0.445 \tag{4.2}$$

The distribution is geometric, so the expected value of the number of iterations is:

$$\mathbb{E}_{\not\Vdash} = \sum_{k=1}^{\infty} kp^{k-1}(1-p) \approx 1.801 \tag{4.3}$$

thus on average less than 2 executions of AES-256 are needed to get to Step 5.

To quantify the probability to meet the condition of Step 5 the size of the database has to be estimated. A very generous upper bound (given the settings that a PAN corresponds to a single client of a Bank) is 10^9 PANs and 10^4 tokens per PAN (generating a new token for every transaction) for a total of 10^{13} entries in the database. Considering that there are $10^{16}-1$ possible tokens, the probability to meet the condition is:

$$\rho = \frac{10^{13}}{10^{16}-1} \approx 0.001$$

Again, the expected value of the number of iterations of the algorithm is:

$$\mathbb{E}_{\not\in} = \sum_{k=1}^{\infty} k\rho^{k-1}(1-\rho) \approx 1.001.$$

thus very rarely more than one execution of SHA-256 is needed.

Finally the expected value of the total number of executions of AES-256 is:

$$\mathbb{E}_{\not\perp}\mathbb{E}_{\not\in} \approx 1.803.$$

thus on average less than 2 executions of AES-256 are needed to get to Step 6, making the algorithm very efficient.

Chapter 5

The BIX Protocol and Certificates: Decentralizing Certificate Authorities

Digital certificates are the very foundation of online security, assuring the integrity of digital communications. Nowadays however Internet relies on trusted third parties to create and guarantee the validity of these certificates, and this centralization may be hazardous in case of large-scale attacks.

In this chapter a distributed protocol is presented, that decentralizes the role of certificate authorities, and its security will be proven in multiple attack scenarios.

This original work has been presented at the NATO CCDCOE conference CyCon held in Tallinn, in May-June 2017, and published in the proceedings [57]. It is a joint work with Dr. Federico Pintore, Dr. Giancarlo Rinaldo and Prof. Massimiliano Sala. The graduand identified the attack scenarios and proved the security of the protocol against static adversaries and collaborated with the coauthors finding the viable attack. Moreover the coauthors helped in the analysis of the protocol, refined and gave structure and substance to the paper.

Introduction

Blockchain is an emerging technology that is gaining widespread adoption to solve a myriad of problems where the classic centralized approach can be substituted by decentralization. Indeed, centralized computations, albeit efficient, are possible only if there is a Trusted Third Party (TTP) that everybody trusts. Nowadays this is sometimes felt as a limitation and a possible vulnerability.

The general idea behind blockchain technology is that blocks containing information are created by nodes in the network, and that these blocks are both public and cryptographically linked, so that an attacker should be unable to modify them without the users noting the tampering. Also, the information contained in any block comes from the users and any user signs his own information cryptographically. Some examples of blockchain applications can be

found in [86], [6], [35].

A very sensitive security aspect which is usually kept centralized is the issuing of digital certificates, which form the core of a Public Key Infrastructure (PKI). A certificate associates at least a cryptographic public key to some identity and is digitally signed by a TTP. An example is given by X.509 certificates [28], mostly containing RSA public keys, which are widely used in the Internet for establishing secure transactions (e.g., an e-payment with an e-commerce site like Amazon). Since every user of a PKI must trust the Certification Authority (CA), which acts as a TTP, the identity of a web site is checked by verifying the CA's signature via the CA's public key. Note that the identity checking is often performed via a hierarchy of CA's.

In a scenario of (possibly state-sponsored) large-scale cyber attacks, Certificate Authorities may become primary targets because of their strategic role in guaranteeing authentication and security of most of the web resources. Unfortunately, their role becomes a liability if they are compromised in the attack, since it becomes impossible for the attacked infrastructure to distinguish fake servers from real ones. Therefore, there is the need of a PKI protocol which is more resilient to wide cyber attacks and which does not introduce *single points of failure*, such as the CAs.

This is exactly the idea behind the so-called BIX certificates. The BIX protocol aims to distribute the role of the CAs, while preserving the security features. Indeed, the BIX protocol is designed with a blockchain-like structure that provides integrity to data, showcasing the possibility of a distributed PKI. A certificate is a block in a blockchain and a valid user interacting properly with the protocol will be able to attach their certificate to the blockchain. The protocol works with very few assumptions on the underlying network, but the original paper by Sead Muftic [63] focuses on the innovative ideas and the technology behind them, leaving formal proofs of security as a stimulating open research problem. In this chapter, after a recapitulation of the BIX protocol, its security is proven, providing suitable formal models of the threat scenarios.

To be more precise, the first attack scenario considered supposes that an attacker tries to attach a new certificate to a pre-existing certificate chain *without* interacting properly with the protocol. This is equivalent to having a malicious user trying to forge a valid certificate for themself (or forge a certificate to assume the identity of an innocent user). The second attack scenario considers that an adversary tries to modify an existing chain of certificates, and distribute it as a valid chain. Finally, an attack is identified where two colluding members of the BIX community can alter an existing chain of certificates.

In Section 5.1 a sketch of Muftic's scheme is provided, highlighting its characteristics that are instrumental in its security.

In Section 5.2 and Section 5.3, the threat scenarios and their actors are formalized, stating their capabilities and goals, in order to build realistic models of the attacks, suitable to undergo formal analyses.

In Section 5.4 an attack on the BIX protocol is shown, which can alter an existing chain when two members of the BIX community collude.

Finally, in Section 5.5 draw our conclusions investigating BIX protocol's resiliency against large-scale cyber attacks.

5.1 Background: A description of BIX certificates

In this section BIX certificates are described, alongside the structure containing them, called the BIX Certification Ledger (BCL). BIX certificates share many similarities with X.509 certificates, for a detailed comparison see [63], Section 2.1; in this section only the characteristics instrumental for the security proofs are highlighted.

5.1.1 Bix Certification Infrastructure (BCI)

Muftic bases his protocol on the BIX Certification Infrastructure (BCI), that is the collection of all BIX certificates issued to BIX members. Because there are no third parties involved, the entities managing certificates are BIX members themselves. This means that members have two roles: as users of the infrastructure and also as certifying and validation authorities. The BCI is a public certificates ledger. It is a double-linked linear list of certificates without branches. Each certificate points to the first previous certificate ("backward" link) that belongs to the user that issued the certificate, and it also points to the next certificate ("forward" link) that was issued by the user that owns this certificate. The BCI starts with the Root Certificate, that is issued by the entity initiating the BCI (equivalent to the genesis transaction in the Bitcoin system). When the Root Certificate is generated, the first user may be registered and their certificate will be issued by the BCI's initiating entity. The last BIX member who joined the system is added to the "tail" of the BIX Certification Ledger and they will be the issuer of the next certificate. The BIX Certificates Ledger can be traversed backwards (to reach the Root Certificate) and forward to find the Issuer for the next certificate. The BCI requires as the operational prerequisite a broadcast messaging system with instantaneous delivery of messages. This system is not a third party, as it only passively distributes BIX certificates and (for addressing purposes) verifies that the BIX Identifier of the new user is unique. The same system is needed for distributed file storage [86]. In Muftic's implementation of the BCI, the secure IM protocol is used for this purpose[87].

5.1.2 The Chain of Certificates

The BCL collects all the BIX certificates filling a double-linked list, in which every certificate is linked to the previous and the next. Let the BCL be a "chain of certificates" CC comprised of n certificates, that may be considered a sequence:

$$\mathrm{CC} : c_0, \ldots, c_{n-1}.$$

To simplify the notation, let λ be a function returning the length of a chain, that is $\lambda(\mathrm{CC}) = n$. Also, $||$ denotes string concatenation.

Remark 5.1. The owner of the certificate c_i has a double role: a **user**, with the certificate c_i that certificates their identity; a **issuer**, providing the certificate c_{i+1} to the next user. In this way there is no need of a CA (Certification Authority).

	Header(H_i)	
	Sequence number Version, Date	
Issuer(S_{i-1})	Subject(S_i)	Next Subject(S_{i+1})
Bix ID of S_{i-1} Public key (PK_{i-1})	Bix ID of S_i Public key (PK_i)	Bix ID of S_{i+1} Public key (PK_{i+1})
Issuer Signature	Subject Signature	Next Subject Signature
Backward cross-signature		
Signature of $(H_i\|\|h(S_{i-1})\|\|h(S_i))$ by SK_{i-1} Signature of $(H_i\|\|h(S_{i-1})\|\|h(S_i))$ by SK_i		
	Forward cross-signature	
	Signature of $(H_i\|\|h(S_i)\|\|h(S_{i+1}))$ by SK_i Signature of $(H_i\|\|h(S_i)\|\|h(S_{i+1}))$ by SK_{i+1}	

Table 5.1: Structure of a BIX certificate

To each certificate corresponds a user having a key-pair private key/public key, denoted with SK_i and PK_i. The certificate c_0 is called the `root certificate` and the certificate c_{n-1} is called the `tail certificate`.
In this thesis a certificate c_i for $i = 1, \ldots, n-2$ is defined by the following fields (and subfields), while the complete list can be found in [63]. Root and tail certificates are described later on.

Header (H_i) in this field there is general information such as timestamps and protocol version, but the relevant information for the present analysis is the **Sequence number** i, that is the identification number of the certificate that is also the position with respect to certificates of other BIX members.

Subject (S_i) The `Subject` contains the personal information that identifies the i-th user (S_i), in particular

- **Subject BIX ID** This is the unique global identifier of the user who owns the certificate. In particular, all BIX ID's contained in the `Subject` fields of a valid chain are distinct.
- **Public key** The cryptographic public key of the owner of the certificate PK_i.

Subject signature It contains the signature over the Subject attributes via the private key SK_i associated to PK_i.

Issuer (S_{i-1}) The `Issuer` field enjoys the same attribute structure of the `Subject` field, but it identifies the BIX member who certified S_i, i.e., it contains the `Subject` attributes of c_{i-1}, which identifies S_{i-1} (the previous member in the BCI).

Issuer signature This field contains the signature over the Issuer attributes created by the Issuer, that is, performed with the private key SK_{i-1} associated to PK_{i-1}.

Backward cross-signature The `Backward_Cross_Signature` contains two signatures, one created by the Issuer S_{i-1} and the other created by the Subject S_i, over the same message: the concatenation of the Header H_i, the hash of the Issuer $h(S_{i-1})$ and the hash of the Subject $h(S_i)$.

Note that this field guarantees validity of the Header and binding between the Subject and the Issuer.

Next Subject (S_{i+1}) The `Next_Subject` field enjoys the same attribute structure of the `Subject` field, but it identifies the BIX member who is certified by S_i, i.e., it contains the `Subject` attributes of c_{i+1}, which identifies S_{i+1} (the next member in the BCI).

Next Subject signature This is the same field as Subject signature, except it is created by the Next Subject over its own data, that is, performed with the private key SK_{i+1} associated to PK_{i+1}.

Forward cross-signature The `Forward_Cross_Signature` contains two signatures, one created by the Subject S_i and the other created by the Next Subject S_{i+1}, over the same message: the concatenation of the Header H_i, the hash of the Subject $h(S_i)$ and the hash of the Next Subject $h(S_{i+1})$.

Note that this field guarantees binding between the current user acting as an issuer and the next user (to whom the next certificate c_{i+1} is issued)

Concerning the special certificates:

- The certificate c_0, called *root certificate*, has the same structure of a standard certificate, but the `Issuer` field and the `Subject` field contain the same data. Indeed, the root user S_0 is not a normal user but rather an entity that initiates the specific BCL.
- Also the certificate c_{n-1} is special. Although it has the same structure of a standard certificate, some fields are not populated because the next user is still unknown: `Next_Subject` , the `Next_Subject` signature, the `Forward_Cross_Signature` . However, note that it is regularly published in the chain and considered valid by other users.

  The last user that owns the last certificate, c_{n-1} will then become the issuer for the next certificate (see 5.1).

In the BIX protocol a new user requests the issuing of a new certificate through a query to the BIX community, which is processed only by the user that owns the tail certificate of the chain. Furthermore, when two users want to perform a secure communication/transaction they exchange their certificates (contained in the same chain) and verify them.

1. Certification request.
   (a) Let Bob be a new user, who will be S_n, that wants a certificate, so registers himself to the system (BCI). The system provides him the BIX-ID. Then, the user creates his private and public key, creates and signs his `Subject` and sends both as a request to the system. Since he does not know who is the last user, he sends his request to every user of the BCL.

(b) Let Alice be the owner of the tail certificate c_{n-1}, namely S_{n-1}, so she processes this request. That is, she fills the `Issuer` field and the Issuer Signature field of c_n, while creating also an intermediate version of the `Backward_Cross_Signature` field with her private key SK_{n-1}. That is, she signs $\big(H_n||h(S_{n-1})||h(S_n)\big)$ (where "||" is concatenation of strings) and puts it into c_n.

(c) At the same time, she updates her BIX certificate c_{n-1} filling the `Next_Subject` field and the Next Subject Signature field using the data of user S_n. Moreover, she creates an intermediate version of the `Forward_Cross_Signature` field by signing it with her private key. That is, she signs $\big(H_{n-1}||h(S_{n-1})||h(S_n)\big)$ with SK_{n-1}.

(d) Now S_{n-1} sends three certificates, c_0, c_{n-1} and c_n to the new user S_n through the system (BCI). Observe that the two certificates c_{n-1} and c_n are still incomplete and that c_n will be the new tail certificate.

(e) User S_n (Bob) receives these certificates, completes the countersignature process by performing two digital signatures and adding them, respectively, to the `Forward_Cross_Signature` of c_{n-1} and the `Backward_Cross_Signature` of c_n.

(f) User S_n requests the chain CC and checks its integrity, by either traversing it forwards from c_0 to c_{n-1} or backwards from $n-1$ to 0 using c_{n-1}. If CC passes the integrity checks, he broadcasts c_{n-1} and c_n.
At the same time, he stores CC locally for future use.

2. Certificate exchange.
When a user S_i wants to perform a secure communication/transaction with a second user S_j, user S_i checks the certificate c_j in two steps:

(a) verifies the Subject signature, the Issuer signature and also the `Backward_Cross_Signature` of the certificate c_j.

(b) S_i verifies that c_j is in CC.

For further details about the BIX protocol see Section 3.3 of [63].

5.2 Chain Lengthening Attack Scenario

The first attack scenario considered supposes that an attacker tries to attach a new certificate to a pre-existing certificate chain without interacting properly with the last user of the chain. More precisely, the attacker $\mathcal{A}$ should not interact with the owner of the last certificate in the chain according to the BIX protocol.

Definition 5.1 (Static Chain Lengthening (SCL) Game)**.** In this game an adversary $\mathcal{A}$ aims to add a certificate to the tail of a certificate chain CC.
It proceeds as follows:

- The challenger $\mathcal{C}$ builds a certificate chain CC according to the BIX protocol with root certificate c_0, using a hash function h and a digital signature scheme $\mathcal{DSS}$.
- $\mathcal{C}$ passes to $\mathcal{A}$ the complete chain CC together with h and $\mathcal{DSS}$.

- $\mathcal{C}$ builds an honest verifier $\mathcal{V}$ that given a certificate c^* and a certificate chain CC^*, outputs True if the root certificate of CC^* is c_0 and c^* is a valid certificate of CC^*, False otherwise.

- $\mathcal{A}$ tries to build a *forged* certificate chain CC', $\lambda(\mathrm{CC}') = n+1$, such that:

  - CC' truncated before the last certificate c'_n is identical to CC, if the `Next_Subject` and `Forward_Cross_Signature` fields of the second-to-last certificate of CC' are not considered (i.e. CC'can be obtained adding a certificate to CC and completing c_{n-1} accordingly);
  - user S_{n-1} did not take part in the creation of c'_n and so in particular she did not perform the `Forward_Cross_Signature` of c_{n-1} and the `Backward_Cross_Signature` of c'_n;
  - $\mathcal{V}(c', \mathrm{CC}') = \mathsf{True}$ where c'_n is the last certificate of CC'.

$\mathcal{A}$ wins the *SCL* game if CC is successfully lengthened, i.e. if $\mathcal{A}$ builds a CC' that satisfies these last three points.

Definition 5.2 (Security against SCL)**.** The BIX protocol is said *secure against static chain lengthening* if there is no adversary $\mathcal{A}$ that in polynomial time wins the SCL game 5.1 with non-negligible probability.

Theorem 5.1. *Let $\mathcal{A}$ be an adversary that wins the SCL game 5.1 with probability ϵ, then a simulator $\mathcal{S}$ might be built that, with probability at least ϵ, either solves the Collision Problem 2.9, with L the set of all possible `Subject` fields, or wins the Digital Signature Security game 2.7.*

Proof. Let $\mathcal{DSS}$ be the digital signature scheme and h the hash function used in the BIX protocol, and $L \subseteq (\mathbb{F}_2)^l$ be the class of all possible `Subject` fields. We will build a simulator $\mathcal{S}$ that simultaneously plays the Digital Signature Security (DSS) game 2.7 and tries to solve an instance of the Collision Problem 2.9 for L. It does so by simulating an instance of the SCL game 5.1 and exploiting $\mathcal{A}$. It will be proven that if $\mathcal{A}$ wins the SCL game then either $\mathcal{S}$ finds a solution for the Collision Problem or $\mathcal{S}$ wins the DSS game.

$\mathcal{S}$ starts taking as input an instance (h, L) of 2.9 and a public key PK^* given by the $\mathcal{DSS}$ challenger (i.e., the output of the first phase of 2.7 for the scheme $\mathcal{DSS}$). $\mathcal{S}$ then proceeds to build a certificate chain CC^* following the BIX protocol. $\mathcal{S}$ builds all but the last certificate normally, running the KeyGen algorithm of the $\mathcal{DSS}$ to choose public keys for the `Subject` fields, so the corresponding secret keys are available to sign these certificates properly. Then let $n = \lambda(\mathrm{CC}^*) \geq 2$ (i.e. the number of certificates contained in CC^*), c_0^* its root certificate and c_{n-1}^* the last one. $\mathcal{S}$ sets the `Subject` of c_{n-1}^*, that will be denoted by S_{n-1}^*, such that its public key is PK^*, then it queries the challenger of the DSS game to obtain three valid signatures, respectively, on:

- the hash $h(S_{n-1}^*)$ of this subject,
- $\left(H_{n-1}^* || h(S_{n-2}^*) || h(S_{n-1}^*)\right)$ for the `Backward_Cross_Signature` of c_{n-1}^*,
- $\left(H_{n-2}^* || h(S_{n-2}^*) || h(S_{n-1}^*)\right)$ for the `Forward_Cross_Signature` of c_{n-2}^*,

where H^*_{n-2} is the `Header` of c^*_{n-2}, H^*_{n-1} is the `Header` of c^*_{n-1}, and $h(S^*_{n-2})$ is the hash of the `Issuer` of c^*_{n-1}, that is the `Subject` of c^*_{n-2}. In this way $\mathcal{S}$ completes a certificate chain CC^* of length n, that it passes to $\mathcal{A}$.

$\mathcal{A}$ responds with a counterfeit chain CC' of length $\lambda(\mathrm{CC}) = n+1$. If CC' is not valid (the chains CC' and CC^* do not correspond up to the n-th certificate, or an integrity check fails) then $\mathcal{S}$ discards this answer and gives up ($\mathcal{S}$ fails).

Otherwise, if the verifier outputs True, the chain CC' is valid. Denote by l' the string $\big(H'_n||h(S'_{n-1})||h(S'_n)\big)$ signed in the `Backward_Cross_Signature` of c'_n (the last certificate of CC') by the private key corresponding to PK^*. There are two cases:

- l' is equal to a message for which $\mathcal{S}$ requested a signature.
  Because of its bit-length, l' may be equal to $l^*_0 := \big(H^*_{n-2}||h(S^*_{n-2})||h(S^*_{n-1})\big)$ or $l^*_1 := \big(H^*_{n-1}||h(S^*_{n-2})||h(S^*_{n-1})\big)$, but not to $h(S^*_{n-1})$. In either case, $l' = l^*_0$ or $l' = l^*_1$, the equality implies that $h(S'_n) = h(S^*_{n-1})$, but the specification of the BIX protocols supposes that different certificates have a different BIX ID in the `Subject` (and we know that CC' is valid). So $S'_{n-1} = S^*_{n-1} \neq S'_n$, because of the BIX ID's, but they have the same hash so $\mathcal{S}$ may submit (S^*_{n-1}, S'_n) as a solution to the Collision Problem.

- l' is different from all messages for which $\mathcal{S}$ requested a signature.
  In the `Backward_Cross_Signature` of c'_n there is a signature s of l' such that $\mathsf{Ver}(l', s, \mathsf{PK}^*) = \mathsf{True}$ (remember that PK^* is the public key of the `Issuer` of c'_n and that CC' is considered valid, so the signatures check out), so $\mathcal{S}$ may submit (l', s) as a winning answer of the challenge phase of the DSS game.

So if $\mathcal{S}$ does not fail, it correctly solves the Collision Problem or wins the DSS game, and since $\mathcal{A}$ is a polynomial-time algorithm, $\mathcal{S}$ is a polynomial-time algorithm too, given that the other operations performed correspond to the building of a certificate chain and this must be efficient. $\mathcal{S}$ might fail only if the chain given by $\mathcal{A}$ is not valid (i.e. if $\mathcal{A}$ fails). Since the simulation of the SCL game is always correct, $\mathcal{A}$'s failure happens with probability $1-\epsilon$, then the probability that $\mathcal{S}$ wins is $1-(1-\epsilon) = \epsilon$. □

Corollary 5.1 (SCL Security)**.** *If the Digital Signature Scheme is secure (Assumption 2.8) and the hash function is collision resistant for the class L (Assumption 2.10), where L is the set of all possible* `Subject` *fields, then the BIX protocol is secure against the Static Chain Lengthening.*

Proof. Thanks to Theorem 5.1, given a polynomial-time adversary that wins the SCL game with non-negligible probability ϵ, a polynomial-time simulator might be built that with the same probability either solves the Collision Problem 2.9 or wins the Digital Signature Security game 2.7. So let C be the event "solution of the Collision Problem" and D be the event "victory at the DSS game". Then the probability to solve at least one problem is:

$$\epsilon = P(C \vee D) \leq P(C) + P(D) \tag{5.1}$$

Note that the sum of two negligible quantities is itself negligible, so the fact that ϵ is non-negligible implies that at least one of $P(C)$ and $P(D)$ is non-negligible, and this means that Assumption 2.10 or Assumption 2.8 is broken. □

Let Alice be the user of the second-to-last certificate in the chain and Bob the user of the last certificate. Note that the infeasibility of the attack above guarantees also the non-repudiation property of the last certificate in the chain. That is, if Alice tries to repudiate Bob (with an eye to issuing another certificate), then Bob might claim his righteous place showing a version of the chain containing his certificate. This chain is then the proper one since no one can attach its certificate to the tail of the certificate chain without properly interacting with the protocol and being a legitimate user.

In the proof it is assumed (see Assumption 2.8) that $\mathcal{DSS}$ is secure against existential forgery, but to be precise this assumption can be mildly relaxed. In fact in the proof the freedom of the attacker in the choice of the message to be signed is limited. To be precise $\mathcal{A}$ has to forge a signature of $l' := \left(H'_n||h(S'_{n-1})||h(S'_n)\right)$, where $h(S'_{n-1})$ is given by $\mathcal{S}$, and even H'_n is not completely controlled by $\mathcal{A}$ (the sequence number is given, and the other fields should be given by the IM). So a large part of the string to be signed is beyond the control of the forger, hence the challenge is not completely an existential forgery but something in between an existential and a universal forgery, which is a weaker assumption on the Digital Signature Scheme ($\mathcal{DSS}$). However the security of $\mathcal{DSS}$ against universal forgery is not sufficient for this proof, so the stronger assumption has been used.

5.3 Certificate Tampering

In the second attack scenario considered, a malicious attacker tries to corrupt a chain of certificates built upon a trusted root certificate, resulting in another chain that may be re-distributed as a proper chain with same root but with altered information.

The security against this attack would guarantee that no external attacker can modify any certificate in the chain, including deleting or inserting a certificate in any non-ending point, as long as the root certificate is safe (no unauthorized usage), secure (cannot be broken) and public (anyone can check it). If the security proved in the previous section is also considered, then a certificate chain is also secure at the end point (no one can wrongfully insert themself at the end or disavow the last certificate) achieving comprehensive security from external attacks to the BIX protocol.

Definition 5.3 (Static Tampering with Subject (STS) Game)**.** In this game an adversary $\mathcal{A}$ aims to modify information contained in the *Subject* field of a certificate c_i contained in a certificate chain CC, with $1 \leq i \leq n-1$, $n = \lambda(\text{CC})$. It proceeds as follows:

- The challenger $\mathcal{C}$ builds a certificate chain CC with root certificate c_0, according to the BIX protocol and using a hash function h and a Digital Signature Scheme $\mathcal{DSS}$. Let $n = \lambda(\text{CC})$.
- $\mathcal{C}$ passes to $\mathcal{A}$ the complete chain CC together with h and $\mathcal{DSS}$.
- $\mathcal{C}$ builds an honest verifier $\mathcal{V}$ that, given a certificate c^* and a certificate chain CC^*, outputs True if the root certificate of CC^* is c_0 and c^* is a valid certificate of CC^*, False otherwise.

- $\mathcal{A}$ tries to build a forged certificate chain CC$'$ such that:
  - exists $1 \leq i \leq n-1$ such that `Subject` fields of c_i and c'_i are different, that is, $S_i \neq S'_i$.
  - $\mathcal{V}(c', \text{CC}') = \mathsf{True}$

$\mathcal{A}$ wins the *STS* game if he achieves the last two items, i.e. successfully builds such a CC$'$.

Definition 5.4 (Security against STS)**.** The BIX protocol is said *secure against Static Tampering with Subject* if there is no adversary $\mathcal{A}$ that in polynomial time wins the STS game 5.3 with non-negligible probability.

Theorem 5.2. *Let $\mathcal{A}$ be an adversary that wins the STS game 5.3 with probability ϵ, then a simulator $\mathcal{S}$ can be built that with probability at least $\frac{\epsilon}{n-1}$ either solves the collision problem 2.9, where L is the set of all possible `Subject` fields, or wins the Digital Signature Security Game 2.7, where n is the length of the certificate chain that $\mathcal{S}$ gives to $\mathcal{A}$.*

Proof. Let $\mathcal{DSS}$ be the Digital Signature Scheme and h the hash function used in the BIX protocol, and $L \subseteq (\mathbb{F}_2)^l$ be the class of all possible `Subject` fields. A simulator $\mathcal{S}$ will be built that simultaneously plays the digital signature security (DSS) game 2.7 and tries to solve an instance of the collision problem 2.9 for L. It does so by simulating an instance of the STS game 5.3 and exploiting $\mathcal{A}$. It will be proven that when $\mathcal{A}$ wins the STS game, at least one in $n-1$ times $\mathcal{S}$ is successful. To be more precise, if $\mathcal{S}$ does not find a solution for the collision problem then $\mathcal{S}$ wins the DSS game.

$\mathcal{S}$ starts with taking as input an instance (h, L) of the Collision Problem and a public key PK^* given by the $\mathcal{DSS}$ challenger (i.e., the output of the first phase of the Digital Signature Security Game for the scheme $\mathcal{DSS}$).
$\mathcal{S}$ now proceeds to build a certificate chain CC following the BIX protocol, as follows. First, $\mathcal{S}$ chooses $n \geq 2$ (possibly depending on the $\mathcal{A}$'s requirements). Then $\mathcal{S}$ selects $1 \leq k \leq n-1$ at random to be the index of a certificate c_k in CC. $\mathcal{S}$ builds the first $k-1$ certificates normally, running the KeyGen algorithm of the $\mathcal{DSS}$ scheme to choose public keys for the `Subject` fields, so the corresponding secret keys are available (to $\mathcal{S}$) to sign these certificates properly. So $c_0, \ldots, c_{k-3}$ are complete certificate and c_{k-2} is a tail certificate. Then it sets the `Subject` of c_{k-1} such that its public key is PK^*, and a header H_{k-1}. It queries the challenger of the DSS game to obtain three valid signatures, respectively, on:

- the hash $h(S_{k-1})$ of this subject,
- $(H_{k-1}||h(S_{k-2})||h(S_{k-1}))$ for the `Backward_Cross_Signature` of c_{k-1} (if $k > 1$),
- $(H_{k-2}||h(S_{k-2})||h(S_{k-1}))$ for the `Forward_Cross_Signature` of c_{k-2} (if $k > 1$),

where H_{k-2} is the `Header` of c_{k-2}, H_{k-1} is the `Header` of c_{k-1}, $h(S_{k-2})$ is the hash of the `Issuer` of c_{k-1}. Then $\mathcal{S}$ builds the $k+1$-th certificate, choosing a H_k and S_k, using again the KeyGen algorithm to sign S_k, querying the DSS challenger for two valid signatures, respectively, on:

- $(H_k||h(S_{k-1})||h(S_k))$ for the `Backward_Cross_Signature` of c_k,
- $(H_{k-1}||h(S_{k-1})||h(S_k))$ for the `Forward_Cross_Signature` of c_{k-1},

where H_k is the `Header` of c_k and $h(S_k)$ is the hash of the `Subject` of c_k. Finally, $\mathcal{S}$ completes the chain CC (following the protocol and choosing everything, including the SK_i's), so that it has n certificates, and passes it to $\mathcal{A}$.

$\mathcal{A}$ responds with a counterfeit chain CC$'$. $\mathcal{A}$ fails if and only if CC$'$ is not valid, which happens when there is no $1 \leq i \leq n-1$ such that $S'_i \neq S_i$ or when the integrity check of the verifier fails. In this situation, $\mathcal{S}$ discards CC$'$ and gives up ($\mathcal{S}$ fails).

Otherwise, let $1 \leq i \leq n-1$ be the first index for which $S'_i \neq S_i$. Since k is chosen at random we have that $k = i$ with probability $\frac{1}{n-1}$. In this case there are two possibilities:

- $h(S_k) = h(S'_k)$, but $S_k \neq S'_k$ for hypothesis, then $\mathcal{S}$ outputs the pair (S_k, S'_k) as a solution to the collision problem.
- Otherwise, it follows that $S_{k-1} = S'_{k-1}$ and PK^* is the public key of the issuer of c'_k. Then in the `Backward_Cross_Signature` of the certificate c'_k there is the digital signature s for which holds the relation $\mathsf{Ver}\left(\left(H'_k||h(S'_{k-1})||h(S'_k)\right), s, \mathsf{PK}^*\right) = \mathsf{True}$ (remember that CC$'$ is considered valid, so the signatures check out). So $\mathcal{S}$ may submit

$$\left(\left(H'_k||h(S'_{k-1})||h(S'_k)\right), s\right)$$

as a winning answer of the challenge phase of the DSS game, since it is different from the messages $\mathcal{S}$ queried for signatures, that are

$$[\, h(S_{k-1}),\ (H_{k-1}||h(S_{k-2})||h(S_{k-1}))\ ,\ (H_{k-2}||h(S_{k-2})||h(S_{k-1}))\,,$$

$$(H_k||h(S_{k-1})||h(S_k))\,,\ (H_{k-1}||h(S_{k-1})||h(S_k))\ ]\ .$$

So $\mathcal{S}$ correctly solves the collision problem or wins the DSS game at least when $\mathcal{A}$ wins and $i = k$. The probability of this event is at least the probability of the two cases and so it is

$$\epsilon \cdot \frac{1}{n-1} \ = \ \frac{\epsilon}{n-1}\,.$$

Note also that since $\mathcal{A}$ is a polynomial time algorithm, $\mathcal{S}$ is a polynomial time algorithm too. □

Corollary 5.2 (STS Security)**.** *If the Digital Signature Scheme is secure (see Assumption 2.8) and the hash function is collision resistant for the class L (Assumption 2.10) where L is the set of all possible* `Subject` *fields, then BIX protocol is secure against the Static Tampering with Subject.*

Proof. For the BIX protocol to be functional the length of the chain must be polynomial. So, for the result of Theorem 5.2, given a polynomial time adversary that wins the STS game with non-negligible probability ϵ, a polynomial time simulator might be built that with probability at least $\frac{\epsilon}{n-1}$ either solves the Collision Problem 2.9 or wins the Digital Signature Security Game 2.7, where n is the length of the chain. So let C be the event "solution of the Collision

Problem" and D be the event "victory at the DSS game". Then the probability to solve at least one problem is:

$$\frac{\epsilon}{n-1} \leq P(C \vee D) \leq P(C) + P(D) \tag{5.2}$$

Note that the sum of two negligible quantities is itself negligible, so the fact that $\frac{\epsilon}{n-1}$ is non-negligible implies that at least one of $P(C)$ and $P(D)$ is non-negligible, and this means that Assumption 2.10 or Assumption 2.8 is broken. □

5.4 Mid-Chain Altering

The proofs of security in the previous two sections do not show the impregnability of BIX to every attack. In this section an effective attack is presented, where two non-consecutive members of the BIX community (i.e. whose BIX certificates are not next to each other in the chain) collude to create an alternate version of the chain between their two certificates that is considered valid by the members outside of that section, but that contains subjects chosen arbitrarily by the two malicious users.

Let Alice and Bob (S_i, S_j) be two malicious colluding users, where the indexes i, j of their certificates in the certificate chain are such that $j > i+1 > i > 0$. Suppose that the chain is built properly up to the j-th certificate. The claim is that, once Alice (S_j) has received her certificate c_j, she may collude with Bob (S_i) in order to change the information in the certificates c_k with $i < k < j$ in such a way that every user S_m with a certificate with index $0 < m < i$ or $m > j$ will consider correct the altered certificates (if they have not already obtained the original certificates).

The first thing they do is to change the information in the `Subject` fields S_k $(i + 1 \leq k \leq j - 1)$ by generating private keys and the corresponding public ones (and then they are able to sign everything). Then Alice (S_i) changes her certificate c_i so that the fields `Next_Subject` and `Forward_Cross_Signature` link to the altered information and validate it, and similarly does Bob (S_j) with his fields `Issuer` and `Backward_Cross_Signature` of c_j.

At this point this altered version of the chain is considered valid by unsuspecting users. Moreover, Bob as last user is responsible to supply the certificates in the chain to new users, so he may propagate the altered version, while older users S_m with $0 < m < i$ will unwillingly authenticate altered certificates. Indeed, when checking the integrity by traversing the chain either forward or backward, they find no inconsistency as long as Alice points to the altered version.

However, a chain of BIX certificates is a distributed ledger, like the public blockchains of cryptocurrencies (see for example [65] or [35]). So there is an easy mitigation to the highlighted threat scenario, that consists simply in distributing the chain all over the network. In particular it is essential to avoid that a single user is the sole depository of a portion of the chain, thus introducing a single point of failure. So it is sufficient to slightly alter Muftic's original protocol by requiring that, when a new certificate is added to the chain, it has to be broadcast to multiple nodes in the network (similar to *full nodes* of Bitcoin network). Furthermore, when a user intends to start a communication with

another user, they have to request the whole chain (or a portion of it), obtaining it from one of these full nodes randomly chosen. In this way, one has to collude with the majority of the full nodes to reliably spread an altered version of the chain.

Moreover note that if two different (but seemingly equally valid) versions of a certificate are in circulation users can deal with the inconsistency either trusting the older version (following the timestamp of the messaging system), or having a more conservative approach and discard both versions. In the latter case the legitimate user should re-enrol in the system.

5.5 Remarks

In this chapter the BIX certificates protocol proposed in [63] has been formally analyzed from a security point of view. In particular the security against static attacks that aim to corrupt a chain have been proven, reducing the security to the choice of an adequate hash function and digital signature scheme.

The current BIX protocol is still incomplete for it to be considered a full PKI. Possibly, the main lack is the absence of a procedure to revoke or to renew certificates. This is an open problem and further research effort is needed.

However, the sy proofs given show how the BIX infrastructure is a reliable structure for storing public identities in a distributed and decentralized way. While a targeted attack to a CA can result in the issuing of malicious certificates or revocation of valid ones, shattering every certificate it issued, in the case of a cyber attack BIX certificates could still be trusted because no single entity could be targeted and exploited to take down the entire system. Suppose that BIX certificates are issued and distributed in *peacetime*, so that when an emergency breaks out the infrastructure is ready to cope with possible attacks. Indeed, the properties proven in this work guarantee the integrity of the information contained in the certificate chain, so users can rely upon it even in the middle of a cyber attack. In other words, it is true that a targeted offensive to the owner of the last certificate would disrupt the protocol, preventing the issuing of new certificates, nevertheless, if this user is taken down, the validity of existing certificates will still hold.

It may seem that the BIX protocol relies on a Trusted Third Party, the messaging system. However this system is not a third party, as highlighted by Muftic [63], since it only passively broadcasts certificates and, for purely addressing purposes, it verifies the uniqueness of BIX identifiers (in Muftic's construction the IM protocol is used [87]). So a PKI system based on the BIX protocol is more resilient to a wide-scale cyber attack than the standard PKI protocols based on CAs.

Regarding related research, the idea of using a public ledger for digital identities has prominent applications in the distribution of Bitcoin wallet addresses, see for example [19], but there are also applications that try to leverage the functionalities of cryptocurrencies to improve PKI. For example, Matsumoto and Reischuk [60] exploit smart contracts on Ethereum to deter misbehavior of CAs and to incentive extended vigilance over CAs' behavior. However, this approach is not sufficient in case of a large-scale cyber attack, because the financial losses that this solution enforces would affect the attacked CA and not the attackers themselves.

Chapter 6

Public Ledger for Sensitive Data

The rise of Bitcoin has shown to the world the great potential of blockchain technologies. Satoshi Nakamoto's breakthrough allows to build publicly verifiable and almost immutable ledgers, but sometimes privacy has to be factored in.

In this chapter is presented an original protocol that allows sensitive data to be stored on a ledger where its integrity may be publicly verified, but its privacy is preserved.

6.1 Introduction

Public ledgers based on blockchains are a great solution for storing public information and they guarantee immutability and accountability.

However serious problems may arise when they are used to store sensitive data, such as health records. Moreover this kind of data often needs to be shared between multiple service providers (such as healthcare insurance companies, hospitals, pharmacies) possibly with warranties that this data is legit and not been tampered with.

In order to deal with this kind of application an original protocol has been developed that constructs a public ledger that securely and privately stores sensitive information. The intent is to allow secure sharing of this data between authorized parties, and integrate an end-to-end one-time access system that provides full control over the usage of private data.

The approach used to achieve a one-time access system is based on the fact that when a party gains access to a cleartext, it has the opportunity to copy the data and store it locally. If this is the case, it is worthless to negate further access to that cleartext, since that party already owns a copy. However it is often expensive to maintain a local copy of the data, so the goal becomes to avoid further access to the data *unless* a local copy has been made. More precisely the aim is to guarantee that no party has access to information with size smaller than the entire cleartext, that can grant further access to the data (e.g. a key).

So the idea is to encrypt the cleartext with a sort of one-time-pad scheme, in this way the key has at least the same size of the cleartext, so storing it does not

give any advantage over storing the cleartext directly. To achieve practicality the pad is stored encrypted on the ledger, so a smaller key is sufficient to recover the original data. To maintain the one-time property the pad is periodically re-encrypted so a new key is needed to access to the data again.

6.2 Masking Shards Protocol

In this protocol multiple users U_l publish encrypted data on a public ledger maintained by a file keeper F. To gain access to the encrypted data any service provider P has to ask directly to the user for a decryption key to use in combination with the masking shards published on the ledger; thus the encryption is effectively end-to-end. The file keeper periodically updates the masking shards, so that older decryption keys become useless.

The ledger also has a constant section where encrypted data is actually stored and the integrity is guaranteed via chains of hash digests.

Definition 6.1 (Updating Masking Shards Protocol)**.** An *Updating Masking Shards Protocol* for a file keeper F, a set of users $\{U_l\}_{1\leq l\leq N}$ and a service provider S proceeds along the following steps:

- F sets up the public ledger: a bilinear group $\mathbb{G}_1$ of prime order p is chosen according to a security parameter κ, along with a generator $g \in \mathbb{G}_1$. Let e be the pairing and $\mathbb{G}_2$ be the target group of the same order p. F chooses uniformly at random exponents $u_i \in \mathbb{Z}_p$ for $1 \leq i \leq I$ where I is the maximum number of shards in a block, determined from the desired block length $|B|$ by the formula $I = |B|/\delta$ where δ is the number of bits required to represent an element of $\mathbb{G}_2$. Finally F chooses a random time-key $s_{t_0} \in \mathbb{Z}_p$ and publishes the initial masking shards:

$$\varepsilon_{i,t_0} = g^{u_i s_{t_0}} \qquad 1 \leq i \leq I \tag{6.1}$$

  F securely saves the value s_{t_0} but can forget the exponents u_i.

- F periodically updates the shards. choosing a new time-key $s_{t_{j+1}} \in \mathbb{Z}_p$ and computing

$$\varepsilon_{i,t_{j+1}} = (\varepsilon_{i,t_j})^{\frac{s_{t_{j+1}}}{s_{t_j}}} \qquad 1 \leq i \leq I \tag{6.2}$$

- Each user U_l chooses two private exponents $\mu_l, v_l \in \mathbb{Z}_p$ and publishes their public key:

$$q_l = g^{\mu_l} \tag{6.3}$$

- To publish an encrypted file on the ledger at a time t_j, a user U_l requests an encryption token. F takes the public key q_l of the user and computes:

$$\begin{aligned} k_{l,0,t_j} &= q_l^{\frac{1}{s_{t_j}}} \\ &= g^{\frac{\mu_l}{s_{t_j}}} \end{aligned} \tag{6.4}$$

- Let $b-1$ be the index of the last block in the ledger, thus the file will be published in the b-th block. Let m_b be the message (file) that U_l wants to encrypt, and $I_b\delta$ its length. Then U_l divides the message in pieces of equal length δ (padding if necessary), and computes their digest through a secure hash function h:

$$h(m_{b,i}) \qquad 1 \leq i \leq I_b \tag{6.5}$$

Then U_l chooses a random exponent $k_b \in \mathbb{Z}_p$ and computes the encrypted shards as:

$$\begin{aligned} c_{b,i} &= m_{b,i} \oplus e(\varepsilon_{i,t_j}, (k_{l,0,t_j})^{k_b}) \qquad 1 \leq i \leq I_b \\ &= m_{b,i} \oplus e(g^{u_i s_{t_j}}, g^{\frac{k_b \mu_l}{s_{t_j}}}) \\ &= m_{b,i} \oplus e(g,g)^{u_i k_b \mu_l} \end{aligned} \tag{6.6}$$

F inserts these encrypted shards and digests into the next block of the static public chain.

- In addition to the encrypted shards U_l computes the encapsulated key:

$$\begin{aligned} k_{b,1,t_j} &= (k_{l,0,t_j})^{\frac{v_l k_b}{\mu_l}} \\ &= (g^{\frac{\mu_l}{s_{t_j}}})^{\frac{v_l k_b}{\mu_l}} \\ &= g^{\frac{v_l k_b}{s_{t_j}}} \end{aligned} \tag{6.7}$$

and F inserts it in the updating ledger. U_l can forget the exponent k_b once this key has been computed and given to F.

- Let $\bar{i} = b \mod I$. Then F computes the control shard:

$$\begin{aligned} c_b &= e(\varepsilon_{\bar{i},t_j}, k_{b,1,t_j}) \\ &= e(g^{u_{\bar{i}} s_{t_j}}, g^{\frac{k_b v_l}{s_{t_j}}}) \\ &= e(g,g)^{u_{\bar{i}} k_b v_l} \end{aligned} \tag{6.8}$$

At this point the block is completed computing the hash digest of its content.

- Once that at least one file has been published F has to periodically update not only the masking shards but also the encapsulated keys. The key update is similar to the shard update:

$$\begin{aligned} k_{b,1,t_{j+1}} &= (k_{b,1,t_j})^{\frac{s_{t_j}}{s_{t_{j+1}}}} \\ &= (g^{\frac{v_l k_b}{s_{t_j}}})^{\frac{s_{t_j}}{s_{t_{j+1}}}} \\ &= g^{\frac{v_l k_b}{s_{t_{j+1}}}} \end{aligned} \tag{6.9}$$

Where $s_{t_{j+1}} \in \mathbb{Z}_p$ is the same time-key used to update the shards. Note that s_{t_j} could and should be forgotten once the update has been completed.

- Let P be a service provider that needs access to the file m_b, and therefore asks for permission to the owner of the file U_l. To grant a one-time permission (or better permission until the next update) U_l computes an unlocked key valid for the current time t_j. U_l retrieves from the updating ledger the encapsulated key $k_{b,1,t_j}$ and computes:

$$\begin{aligned} k_{b,2,t_j} &= (k_{b,1,t_j})^{\frac{\mu_l}{v_l}} \\ &= (g^{\frac{v_l k_b}{s_{t_j}}})^{\frac{\mu_l}{v_l}} \\ &= g^{\frac{\mu_l k_b}{s_{t_j}}} \end{aligned} \tag{6.10}$$

- With the unlocked key P can decrypt the encrypted shards computing:

$$\begin{aligned} m'_{b,i} &= c_{b,i} \oplus e(\varepsilon_{i,t_j}, k_{b,2,t_j}) \qquad 1 \leq i \leq I_b \\ &= m_{b,i} \oplus e(g,g)^{u_i k_b \mu_l} \oplus e(g^{u_i s_{t_j}}, g^{\frac{\mu_l k_b}{s_{t_j}}}) \\ &= m_{b,i} \oplus e(g,g)^{u_i k_b \mu_l} \oplus e(g,g)^{u_i \mu_l k_b} \\ &= m_{b,i} \end{aligned} \tag{6.11}$$

6.3 Block structure

The file keeper collects encrypted data from the various users and organizes it into blocks. The ledger has a variable part that contains at any time t:

- the masked shards and their index

$$(\varepsilon_{i,t}, i)_{1 \leq i \leq I}$$

- the encapsulated keys and the index of the block where the corresponding encrypted pieces are stored

$$(k_{b,1,t}, b)_{b \geq 1}$$

All these elements are kept constantly updated. The other part is comprised of constant blocks chained together. Each block B_b contains:

- the encrypted shards of the file, the digests of the cleartext shards, and their index:

$$(c_{b,0i}, h(m_{b,i}), i)_{1 \leq i \leq I_b}$$

- the control shard c_b
- the hash of the previous block $h(B_{b-1})$
- the hash digest of the encrypted shards, the digests, the control shard and the digest of the previous block:

$$d_b = h(c_{b,1}||h(m_{b,1})|| \ldots ||c_{b,I_b}||h(m_{b,I_b})||c_b||h(B_{b-1})) \tag{6.12}$$

- ▷ the signature of the user (owner of the data) on d_b

- ▷ a proof of work involving d_b
- ▷ a signature made by a third party (or a group or multi-party signature) on d_b

Note that by design the index of the masking shard associated to the control piece covers the whole range $1 \leq i \leq I$. These pieces are needed to check the integrity of the data stored in the updating part of the ledger (encapsulated keys and masking shards). In fact let $\bar{i} = b \mod I$, then for every time t it should hold:

$$\begin{aligned} c_b &= e(\varepsilon_{\bar{i},t}, k_{b,1,t}) \\ &= e(g^{u_{\bar{i}} s_t}, g^{\frac{k_b v_l}{s_t}}) \\ &= e(g,g)^{u_{\bar{i}} k_b v_l} \end{aligned} \tag{6.13}$$

The last three items in the list (marked with a ▷ bullet) represent different approaches to guarantee the immutability of the static block. These solutions are optional and all have pros and cons to their adoption, the optimal choice is probably a combination of the three.

The user signature would give proof of ownership of the data, on the other hand without the signature the content remains fully anonymous to anyone besides F, that can act as a proxy forwarding the access request of P to the right user U_l. To guarantee the end-to-end property of the protocol P should have a public key known to (or retrievable by) every user. So when F forwards a request by P to U_l, the user can encrypt the unlocked key with this public key, so that only P can use it.

The proof of work would give increasingly stronger proof of integrity to older data, in the sense that the more blocks are added to the ledger, the more infeasible it gets to manipulate older data maintaining the consistency of the chain.

The third party signature gives proof of integrity of the block, in alternative or addition to the proof of work. The trade-off between these solutions is that a signature gives instantaneous integrity evidence but relies on the honesty of the signer (possibly mitigated with multi-party signatures), whereas a proof of work is more expensive and provides sufficiently reliable security only for older blocks, but does not rely on anyone's honesty.

In alternative to the digest defined in 6.12, a more efficient blockchain may be built excluding the actual encrypted data from the blocks, retaining only their digests. That is the bulk of data is stored in distributed databases, while their hash is kept on the ledger to guarantee the integrity. This approach reduces consistently the size of the blocks, that therefore can be more widely distributed. This shrunk block would then contain:

- the hash digest of the encrypted shards of the file and the digests of the cleartext shards:

$$d_{b,0} = h(c_{b,1}||h(m_{b,1})|| \ldots ||c_{b,I_b}||h(m_{b,I_b}))$$

- the control shard c_b

- the hash of the previous block $h(B_{b-1})$
- the hash digest of the previous data:

$$d_{b,1} = h(d_{b,0}||c_b||h(B_{b-1}))$$

6.4 Security Model

The goals of the protocol is to achieve the following security properties:

End-to-end encryption The file keeper must not be able to read the plaintext message at any time.

One-time access A service provider should be able to read a plaintext message at the time t if and only if the file owner authorizes them with an unlocked key for the time t.

6.4.1 Security against Outsiders and Service Providers

The security of the protocol will be proven in terms of chosen-plaintext indistinguishability, the security game is formally defined as follows.

Definition 6.2 (Security Game)**.** The security game for an updating masking protocol proceeds as follows:

Init The adversary $\mathcal{A}$ chooses a number of users N that will encrypt files and the maximum number of masking shards I.

Setup For each userU_l, with $1 \leq l \leq N$ the challenger $\mathcal{C}$ sets up a public key q_l, and takes the role of the file keeper by publishing the initial masking shards ε_{i,t_0}, for $1 \leq i \leq I$.

Phase 0 The adversary may request updates of the masking shards ε_{i,t_j}, for $1 \leq i \leq I$, $0 \leq j < n$.

Commit The users commit to a key before creating a ciphertext by publishing the encryption tokens $k_{l,0,t_n}$ and the encapsulated keys $k_{b,1,t_n}$.

Phase 1 The adversary for each time $n \leq j < n^*$ may request updates of the encapsulated keys $k_{b,1,t_j}$, and either the corresponding unlocked keys $k_{b,2,t_j}$, or the masking shards ε_{i,t_j}, but not both.

Challenge Let δ be the number of bits required to represent an element of $\mathbb{G}_2$, thus $\delta = |e(g,g)|$. For each $1 \leq l \leq N$ the adversary chooses two messages $m_{l,0}, m_{l,1}$ of length $I_l\delta$ and sends them to the challenger that flips N random coins $r_l \in \{0,1\}$ and publishes the encryption $(c_{b,i}, 1 \leq i \leq I_l)$ of the message m_{l,r_l}.

Phase 2 Phase 1 is repeated for $n* \leq j < n'$.

Guess The adversary chooses an $\bar{l}$ such that $1 \leq \bar{l} \leq N$ and outputs a guess $r'_{\bar{l}}$ of $r_{\bar{l}}$.

Definition 6.3 (Updating Masking Security)**.** An Updating Masking protocol with security parameter ξ is CPA secure if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function ϕ such that:

$$Pr[r'_{\bar{l}} = r_{\bar{l}}] \leq \frac{1}{2} + \phi(\xi) \tag{6.14}$$

The scheme is proved secure under the BDH assumption (Definition 2.3) in the security game defined above.

The security is provided by the following theorem.

Theorem 6.1. *If an adversary can break the scheme, then a simulator can be constructed to play the decisional BDH game with non-negligible advantage.*

Proof. Suppose there exists a polynomial-time adversary $\mathcal{A}$, that can attack the scheme in the Selective-Set model with advantage ϵ. Then a simulator $\mathcal{B}$ can be built that can play the Decisional BDH game with advantage $\epsilon/2$. The simulation proceeds as follows.

Init The adversary chooses the number of users N and the maximum number of masking shards I, the simulator takes in a BDH challenge $g, A = g^a, B = g^b, C = g^c, T$.

Setup The simulator chooses random $\mu'_l, \omega_l, \mu_l \in \mathbb{Z}_p$ for $1 \leq l \leq N$, $u'_i \in \mathbb{Z}_p$ for $1 \leq i \leq I$. Then it implicitly sets:

$$\mu_l := \mu'_l b \qquad v_l := \frac{\omega_l b}{a} \qquad 1 \leq l \leq N, \qquad u_i := u'_i c \qquad 1 \leq i \leq I \tag{6.15}$$

So it publishes the public keys of the users and the initial masking shards:

$$\begin{aligned} q_l &= B^{\mu'_l} \qquad 1 \leq l \leq N, \\ &= g^{b\mu'_l} \\ &= g^{\mu_l} \end{aligned} \qquad\qquad \begin{aligned} \varepsilon_{i,t_0} &= C^{u'_i s_{t_0}} \qquad 1 \leq i \leq I \\ &= g^{cu'_i s_{t_0}} \\ &= g^{u_i s_{t_0}} \end{aligned} \tag{6.16}$$

Phase 0 In this phase the simulator answers to update queries of the masking shards. For each time $0 \leq j \leq n$ it chooses uniformly at random $s_{t_j} \in \mathbb{Z}_p$ and computes:

$$\begin{aligned} \varepsilon_{i,t_j} &= C^{u'_i s_{t_j}} \qquad 1 \leq i \leq I \\ &= g^{cu'_i s_{t_j}} \\ &= g^{u_i s_{t_j}} \end{aligned} \tag{6.17}$$

Commit In this phase users commit to a key before creating a ciphertext, by publishing their encryption tokens and encapsulated keys. Note that each user may commit at a different time, but for simplicity we suppose that they commit all together. Moreover from now on the indexes b and l will be identified, since for the purposes of this proof only one encryption per user is considered.

To simulate the commitment $\mathcal{B}$ chooses uniformly at random $k'_l \in \mathbb{Z}_p$ and implicitly sets $k_l := k'_l a$ for $1 \leq l \leq N$. Furthermore it chooses $s_{t_n} \in \mathbb{Z}_p$, then it can compute:

$$\begin{aligned} k_{l,0,t_n} &= B^{\frac{\mu'_l}{s_{t_n}}}, & k_{b,1,t_n} &= B^{\frac{\omega_l k'_l}{s_{t_n}}} \qquad 1 \leq l \leq N \\ &= g^{\frac{b\mu'_l}{s_{t_n}}} & &= g^{\frac{b\omega_l a k'_l}{a s_{t_n}}} \\ &= g^{\frac{\mu_l}{s_{t_n}}} & &= g^{\frac{v_l k_l}{s_{t_n}}} \end{aligned} \tag{6.18}$$

Phase 1 In this phase the adversary for each time $n \leq j < n^*$ may request updates of the encapsulated keys $k_{b,1,t_j}$, and either the corresponding unlocked keys $k_{b,2,t_j}$, or the masking shards ε_{i,t_j}, but not both. If the adversary asks for the unlocked keys the simulator chooses at random $s'_{t_j} \in \mathbb{Z}_p$ and implicitly sets $s_{t_j} := s'_{t_j} b$. So it can compute

$$\begin{aligned} k_{b,1,t_j} &= g^{\frac{\omega_l k'_l}{s'_{t_j}}}, & k_{b,2,t_j} &= A^{\frac{k'_l \mu'_l}{s'_{t_j}}} \qquad 1 \leq l \leq N \\ &= g^{\frac{b\omega_l a k'_l}{ab s'_{t_j}}} & &= g^{\frac{a k'_l b \mu'_l}{b s'_{t_j}}} \\ &= g^{\frac{v_l k_l}{s_{t_j}}} & &= g^{\frac{k_l \mu_l}{s_{t_j}}} \end{aligned} \tag{6.19}$$

Otherwise, if the adversary asks for the masking shards, it chooses $s_{t_j} \in \mathbb{Z}_p$ and computes

$$\begin{aligned} k_{b,1,t_j} &= B^{\frac{\omega_l k'_l}{s_{t_j}}} \qquad 1 \leq l \leq N, & \varepsilon_{i,t_j} &= C^{u'_i s_{t_j}} \qquad 1 \leq i \leq I \\ &= g^{\frac{b\omega_l a k'_l}{a s_{t_j}}} & &= g^{c u'_i s_{t_j}} \\ &= g^{\frac{v_l k_l}{s_{t_j}}} & &= g^{u_i s_{t_j}} \end{aligned} \tag{6.20}$$

Challenge For each $1 \leq l \leq N$ the adversary sends two messages $m_{l,0}, m_{l,1}$of length $I_l \delta$. The simulator flips N random coins $r_l \in \{0,1\}$ then creates the ciphertexts as:

$$\begin{aligned} c_{b,i} &:= m_{l,r_l,i} \oplus T^{u'_i k'_l \mu'_l} \\ &\stackrel{*}{=} m_{l,r_l,i} \oplus e(g,g)^{a k'_l b \mu'_l c u'_i} \\ &= m_{l,r_l,i} \oplus e(g,g)^{k_l \mu_l u_i} \qquad 1 \leq i \leq I_l, \quad 1 \leq l \leq N \end{aligned} \tag{6.21}$$

where the equality $\stackrel{*}{=}$ holds if and only if the BDH challenge was a valid tuple (i.e. T is non-random).

Phase 2 During this phase the simulator acts exactly as in *Phase 1.*

Guess The adversary will eventually select a user $\bar{l}$ and output a guess $r'_{\bar{l}}$ of $r_{\bar{l}}$. The simulator then outputs 0 to guess that $T = e(g,g)^{abc}$ if $r'_{\bar{l}} = r_{\bar{l}}$; otherwise, it outputs 1 to indicate that it believes T is a random group element in $\mathbb{G}_2$.

In fact when T is not random the simulator $\mathcal{B}$ gives a perfect simulation so it holds:

$$Pr\left[\mathcal{B}\left(\vec{y}, T = e(g,g)^{abc}\right) = 0\right] = \frac{1}{2} + \epsilon$$

On the contrary when T is a random element $R \in \mathbb{G}_2$ the messages m_{r_l} are completely hidden from the adversary point of view, so:

$$Pr\left[\mathcal{B}\left(\vec{y}, T = R\right) = 0\right] = \frac{1}{2}$$

Therefore, $\mathcal{B}$ can play the decisional BDH game with non-negligible advantage $\frac{\epsilon}{2}$. □

6.4.2 Security Against the File Keeper

In this section we prove the *end to end* privacy of the protocol testing its robustness in scenarios where the File Keeper itself tries to read the content of the encrypted data stored on the ledger. The security of the protocol will again be proven in terms of chosen-plaintext indistinguishability, but there is a distinction between two scenarios.

In the first one the security will be proven using the standard Decisional Bilinear Diffie Hellman Assumption, but the File Keeper is assumed to be not malicious and that it takes over the role after shards initialization. That is the protocol is initialized and then the relevant information is passed to the File Keeper that subsequently fulfills its role following the protocol (but trying to decrypt the files).

In the second scenario the File Keeper independently sets up the masking shards and freely interacts with the users, but in this case the interactive assumption IDDH defined in 2.6 is necessary to prove the security.

The security game of the first scenario is formally defined as follows.

Definition 6.4 (Curious File Keeper Security Game)**.** The security game for the protocol with a curious File Keeper proceeds as follows:

Init The adversary $\mathcal{A}$ chooses a number of users N that will encrypt files and the maximum number of masking shards I.

Setup For each userU_l, with $1 \leq l \leq N$ the challenger $\mathcal{C}$ sets up a public key q_l, and initializes the masking shards ε_{i,t_0}, for $1 \leq i \leq I$, giving also s_{t_0} to $\mathcal{A}$.

Commit $\mathcal{A}$ asks the users commit to a key before creating a ciphertext giving them encryption tokens $k_{l,0,t_j}$, $\mathcal{C}$ responds publishing encapsulated keys $k_{b,1,t_j}$.

Challenge Let δ be the number of bits required to represent an element of $\mathbb{G}_2$, thus $\delta = |e(g,g)|$. For each $1 \leq l \leq N$ the adversary chooses two messages $m_{l,0}, m_{l,1}$ of length $I_l\delta$ and sends them to the challenger that flips N random coins $r_l \in \{0,1\}$ and publishes the encryption $(c_{b,i}, 1 \leq i \leq I_l)$ of the message m_{l,r_l}.

Guess The adversary chooses an $\bar{l}$ such that $1 \leq \bar{l} \leq N$ and outputs a guess $r'_{\bar{l}}$ of $r_{\bar{l}}$.

Definition 6.5 (Security with a Curious File Keeper)**.** An Updating Masking protocol with security parameter ξ is CPA secure with a Curious File Keeper if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function ϕ such that:

$$Pr[r'_{\bar{l}} = r_{\bar{l}}] \leq \frac{1}{2} + \phi(\xi) \tag{6.22}$$

The security is provided by the following theorem.

Theorem 6.2. *If an adversary can break the scheme, then a simulator can be constructed to play the decisional BDH game with non-negligible advantage.*

Proof. Suppose there exists a polynomial-time adversary $\mathcal{A}$, that can attack the scheme in the Selective-Set model with advantage ϵ. Then a simulator $\mathcal{B}$ can be built that can play the Decisional BDH game with advantage $\epsilon/2$. The simulation proceeds as follows.

Init The adversary chooses the number of users N and the maximum number of masking shards I, the simulator takes in a DBDH challenge $g, A = g^a, B = g^b$, $C = g^c, T$.

Setup The simulator chooses random $\mu'_l, \omega_l, \mu_l \in \mathbb{Z}_p$ for $1 \leq l \leq N$, $u'_i \in \mathbb{Z}_p$ for $1 \leq i \leq I$, $s_{t_0} \in \mathbb{Z}_p$. Then it implicitly sets:

$$\mu_l := \mu'_l b \qquad v_l := \frac{\omega_l b}{a} \qquad 1 \leq l \leq N, \qquad u_i := u'_i c \qquad 1 \leq i \leq I \tag{6.23}$$

So it publishes the public keys of the users and the initial masking shards:

$$\begin{aligned} q_l &= B^{\mu'_l} \qquad 1 \leq l \leq N, & \varepsilon_{i,t_0} &= C^{u'_i s_{t_0}} \qquad 1 \leq i \leq I \\ &= g^{b\mu'_l} & &= g^{cu'_i s_{t_0}} \\ &= g^{\mu_l} & &= g^{u_i s_{t_0}} \end{aligned} \tag{6.24}$$

Moreover the exponent s_{t_0} is given to the adversary.

Commit In this phase the adversary asks the users to commit to a key before creating a ciphertext, by giving them encryption tokens and obtaining encapsulated keys in return. For every user l the adversary may choose a different time t_{j_l} in which the commitment takes place. To formulate the query $\mathcal{A}$ chooses a random exponent $s_{t_{j_l}}$ and computes the encryption token:

$$\begin{aligned} k_{l,0,t_{j_l}} &= q_l^{\frac{1}{s_{t_{j_l}}}} \\ &= B^{\frac{\mu'_l}{s_{t_{j_l}}}} \\ &= g^{\frac{b\mu'_l}{s_{t_{j_l}}}} \\ &= g^{\frac{\mu_l}{s_{t_{j_l}}}} \end{aligned} \tag{6.25}$$

To simulate the answer $\mathcal{B}$ chooses uniformly at random $k'_l \in \mathbb{Z}_p$ and implicitly sets $k_l := k'_l a$ for $1 \leq l \leq N$. Then it can compute:

$$\begin{aligned} k_{b,1,t_{j_l}} &= (k_{l,0,t_{j_l}})^{\frac{\omega_l k'_l}{\mu'_l}} \qquad 1 \leq l \leq N \\ &= g^{\frac{b\mu'_l}{s_{t_{j_l}}} \frac{\omega_l k'_l a}{\mu'_l a}} \\ &= g^{\frac{b}{a}\omega_l \frac{a k'_l}{s_{t_{j_l}}}} \\ &= g^{\frac{v_l k_l}{s_{t_{j_l}}}} \end{aligned} \tag{6.26}$$

Challenge For each $1 \leq l \leq N$ the adversary sends two messages $m_{l,0}, m_{l,1}$of length $I_l\delta$. The simulator flips N random coins $r_l \in \{0,1\}$ then creates the ciphertexts as:

$$\begin{aligned} c_{b,i} &:= m_{l,r_l,i} \oplus T^{u'_i k'_l \mu'_l} \\ &\stackrel{*}{=} m_{l,r_l,i} \oplus e(g,g)^{a k'_l b \mu'_l c u'_i} \\ &= m_{l,r_l,i} \oplus e(g,g)^{k_l \mu_l u_i} \qquad 1 \leq i \leq I_l, \quad 1 \leq l \leq N \end{aligned} \tag{6.27}$$

Where the equality $\stackrel{*}{=}$ holds if and only if the BDH challenge was a valid tuple (i.e. T is non-random).

Guess The adversary will eventually select a user $\bar{l}$ and output a guess $r'_{\bar{l}}$ of $r_{\bar{l}}$. The simulator then outputs 0 to guess that $T = e(g,g)^{abc}$ if $r'_{\bar{l}} = r_{\bar{l}}$; otherwise, it outputs 1 to indicate that it believes T is a random group element in $\mathbb{G}_2$. In fact when T is not random the simulator $\mathcal{B}$ gives a perfect simulation so it holds:

$$Pr\left[\mathcal{B}\left(\vec{y}, T = e(g,g)^{abc}\right) = 0\right] = \frac{1}{2} + \epsilon$$

On the contrary when T is a random element $R \in \mathbb{G}_2$ the messages m_{r_l} are completely hidden from the adversary point of view, so:

$$Pr\left[\mathcal{B}\left(\vec{y}, T = R\right) = 0\right] = \frac{1}{2}$$

Therefore, $\mathcal{B}$ can play the decisional BDH game with non-negligible advantage $\frac{\epsilon}{2}$. □

During the simulation in this proof the update of the masking shards and encapsulated keys has not been explicitly considered because the adversary has the role of the File Keeper, that is the party in charge of such operations, so the simulator is not involved.

Note also that for the encryption in the challenge phase the simulator does not use neither the masking shards nor the encapsulated key, so the update of these elements is not strictly necessary (although the simulator should request them even without using them just for the sake of a thorough simulation). This is possible only because the simulator controls the initialization of the masking shards, and this limits the possibilities for the attacker (e.g. choosing particular values for s_t).

To consider a more powerful adversary and take account of possible interaction the security game can be modified in this way:

- in the setup phase the simulator does not initialize the masking shards, the adversary has complete control over them
- in the challenge phase the simulator asks for the updated version of the masking shards and the encapsulated keys before computing the encryption, moreover the adversary may ask that the encryptions take place at different times.

Now the security against this more powerful File Keeper can be given with the following theorem.

Theorem 6.3. *If an adversary taking the role of the file keeper can break the scheme, then a simulator can be constructed to play the IDDH game with non-negligible advantage.*

Proof. The proofs is almost identical to the proof of Theorem 6.2, only the following tweaks are necessary:

- the simulator starts initializing the IDDH challenge, obtaining the elements $A = g^a, B = g^b$;
- during the setup phase the simulator does not have to initialize the masking shards so it does not need the element C
- during the challenge phase suppose that the adversary requests at time t_j the encryption of the user l, then the simulator asks the adversary for the value of the updated masking shards $\varepsilon_{i,t_j} \quad 1 \leq i \leq I$ and the updated encapsulated key $k_{b,1,t_j}$. Then the simulator $\mathcal{B}$ interacts with the challenger $\mathcal{C}$ of the IDDH game sending the value

$$\begin{aligned} S &= k_{b,1,t_j}^{\frac{1}{k'_b \omega_l}} \\ &\overset{\bullet}{=} (g^{\frac{b\omega_l k'_b}{s_{t_j}}})^{\frac{1}{k'_b \omega_l}} \\ &= g^{\frac{b}{s_{t_j}}} \\ &= B^{\frac{1}{s_{t_j}}} \end{aligned} \tag{6.28}$$

  $\mathcal{C}$ answers with Z and $\mathcal{B}$ proceeds with the encryption computing

$$\begin{aligned} c_{b,i} &:= m_{l,r_l,i} \oplus e(\varepsilon_{i,t_j}, Z^{k'_l \mu'_l}) \\ &\overset{\bullet}{=} m_{l,r_l,i} \oplus e(g^{u_i s_{t_j}}, Z^{k'_l \mu'_l}) \\ &\overset{*}{=} m_{l,r_l,i} \oplus e\left(g^{u_i s_{t_j}}, (g^{\frac{ab}{s_{t_j}}})^{k'_l \mu'_l}\right) \\ &= m_{l,r_l,i} \oplus e(g,g)^{u_i a k'_l b \mu'_l} \\ &= m_{l,r_l,i} \oplus e(g,g)^{u_i k_l \mu_l} \qquad 1 \leq i \leq I_l \end{aligned} \tag{6.29}$$

  Where the equality $\overset{*}{=}$ holds if and only if in the IDDH challenge the value of the random coin tossed by $\mathcal{C}$ is $r = 0$. Note that the equalities $\overset{\bullet}{=}$ hold if and only if the adversary followed the protocol acting as the File Keeper, however the interaction with $\mathcal{C}$ is valid even if this is not the case, and from the prospective of $\mathcal{A}$ the simulation has the same distribution of an interaction with the real protocol.

The rest of the simulation is identical and the same considerations hold, thus $\mathcal{B}$ has an advantage of $\frac{\epsilon}{2}$ playing the IDDH. □

6.4.3 Security against other Users

In the previous sections the robustness of the protocol against outside attackers and even against the file keeper has been proven. To complete the analysis are now presented some considerations about the remaining party that participates in the protocol: the users.

Hopefully a user interacting with the protocol gains no advantage in breaking other users privacy. In fact a slight modification of the Theorem 6.1 gives this guarantee.

Consider a setting in which the attacker interacts with the protocol as a normal user (thus requests encryption tokens, publishes encapsulated keys and ciphertexts), but then tries to distinguish the encryption of plaintexts of their choice performed by other users. Starting from the security game of Definition 6.2, phases 1 and 2 can be modified removing unlocked key queries but adding queries for encryption tokens given a public key. After that the adversary has given to the challenger an encapsulated key, the update of these keys may be requested with queries just like any other key. In this way a (possibly malicious) user is modelled, that interacts with the file keeper and can observe the chain and its evolution.

The proof of the security in this scenario follows directly from the proof of Theorem 6.1, in fact without the need of simulating unlocked keys the simulator can always choose s_t freely, and thus can follow servilely the protocol when the adversary requests encryption tokens and updates of the related keys.

6.5 Remarks

The protocol presented in this chapter expands the scope of distributed ledgers to include the safe storage of sensitive data.

The approach used to achieve the one-time access property aims to a highly efficient revocation. That is, the concept of masking shards is used to revoke the access to every ciphertext updating only common elements. This means that it is not necessary to update every block, but only the shards and the encapsulated keys, that are very much shorter than the actual data. Furthermore this allows to check the integrity of the ciphertexts even before decryption, and any observer can monitor the integrity of the ledger checking the coherence of the hash digests in the static chain and verifying the control shards against the data contained in the updating section.

Given the proofs of security against a curious file keeper, it follows that this role is only busy updating shards and encapsulated keys, but is not depositary of trust in a privacy sense. To further reduce the dependence on the file keeper, the protocol can be modified in order to employ temporary file keepers, that are only responsible to perform one update. The passage of responsibilities can be done in multiple ways, e.g. choosing a random candidate in a given set, voting the preferred successor. Anyway the current F chooses a random s_t, while the exponent s_{t-1} is obtained from the previous file keeper in a safe and secure

way; once the update has been completed and a successor has been nominated F passes on s_t. Note that this method enforces the oblivion of old exponents, preventing previous file keepers to reverse a revocation.

A final remark regards the frequency of revocation. As presented here the protocol revokes every ciphertext at once, and while this might be convenient in terms of revocation efficiency, not every application suits this approach, in particular when it is not feasible to burden the user with frequent unlocking of encapsulated keys to restore access to revoked files. An easy solution is to employ different sets of encryption shards and dividing the updating section of the ledger according to different frequencies of revocation. For example a practical ledger could have a set of shards updated with medium frequency, suitable for most of the regular files, a set reserved for quick revocation of very sensitive files, and finally a set for long term accesses updated only in case of necessity.

Implementation

This protocol has not been implemented yet, but an approach similar to that of section 3.3.4 could be used to develop a proof-of-concept. Random generation of keys and parameters would be once again a delicate aspect, alongside the choice of parameters for the underlying algebraic structures. An extensive analysis should also be necessary to choose the best method for block validation (digital signature, or proof of work, or proof of stake, or other solutions) and tailor the length of blocks and shards and the update frequency to the specific needs. Finally the heavy usage of pairing computation could have a distinctive impact on the performance of the protocol, making even more important the choice of a bilinear group that allows fast-enough computations.

Chapter 7

A Proof-of-Stake protocol for Consensus on Bitcoin subchains

Although the transactions on the Bitcoin blockchain have the main purpose of recording currency transfers, they can also carry a few bytes of metadata. A sequence of transaction metadata forms a subchain of the Bitcoin blockchain, and it can be used to store a tamper-proof execution trace of a smart contract.

In this chapter a consensus protocol is presented, based on Proof-of-Stake, that incentives nodes to extend the subchain consistently. The security of the protocol is analysed under many aspects and also with empirical methods, furthermore it is shown how to exploit it as the basis for smart contracts on Bitcoin.

This is a joint work with Sebastian Podda, Stefano Lande, prof. Massimo Bartoletti and prof. Massimiliano Sala, part of the work has been presented at the 1st Workshop on Trusted Smart Contracts CyCon held in Malta, in April 2017 [10]. The graduand identified and proved the vulnerability of the refund policy initially chosen and proposed alternatives. Then the graduand collaborated closely with Sebastian Podda to analyse the various policies, to develop and to prove the effectiveness of the *harsh policy*.

7.1 Introduction

Recently, cryptocurrencies like Bitcoin [65] have pushed forward the concept of decentralization, by ensuring reliable interactions among mutually distrusting nodes in the presence of a large number of colluding adversaries. These cryptocurrencies leverage on a public data structure, called *blockchain*, where they permanently store all the transactions exchanged by nodes. Adding new blocks to the blockchain (called *mining*) requires to solve a moderately difficult cryptographic puzzle. The first miner who solves the puzzle earns some virtual currency (some fresh coins for the mined block, and a small fee for each transaction included therein). In Bitcoin, miners must invert a hash function whose complexity is adjusted dynamically in order to make the average time to

solve the puzzle ∼10 minutes. Instead, removing or modifying existing blocks is computationally unfeasible: roughly, this would require an adversary with more *hashing power* than the rest of all the other nodes. If modifying or removing blocks were computationally easy, an attacker could perform a *double-spending* attack where some amount of coins are paid to a merchant (by publishing a suitable transaction in the blockchain) and then, after the item has been received, the attacker removes the block containing the transaction. According to the folklore, Bitcoin would resist to attacks unless the adversaries control the majority of total computing power of the Bitcoin network. Even though some vulnerabilities have been reported in the literature(see Section 7.5), in practice Bitcoin has worked surprisingly well so far: indeed, the known successful attacks to Bitcoin are standard hacks or frauds [39], unrelated to the Bitcoin protocol.

The idea of using the Bitcoin blockchain and its consensus protocol as foundations for *smart contracts* (namely, decentralized applications beyond digital currency [81]) has been explored by several recent works. For instance, [4, 9, 12, 17, 45, 46, 47] propose protocols for secure multiparty computations and fair lotteries; [29] implements decentralised authorization systems on Bitcoin, [74, 82] allow users to log statements on the blockchain; [21] is a key-value database with get/set operations; [30] extends Bitcoin with advanced financial operations (like e.g., creation of virtual assets, payment of dividends, *etc.*), by embedding its own messages in Bitcoin transactions.

Although the Bitcoin blockchain is primarily intended to trade currency, its protocol allows clients to embed a few extra bytes as metadata in transactions. Many platforms for smart contracts exploit these metadata to store a persistent, timestamped and tamper-proof historical record of all their messages [2, 11]. Usually, metadata are stored in `OP_RETURN` transactions [3], making them meaningless to the Bitcoin network and unspendable. With this approach, the sequence of platform-dependent messages forms a *subchain*, whose content can only be interpreted by the nodes that execute the platform (referred as *meta-nodes*, to distinguish them from Bitcoin nodes). However, since the platform logic is separated from the Bitcoin logic, a meta-node can append to the subchain transactions with metadata which are meaningless for the platform — or even *inconsistent* with the intended execution of the smart contract. However, it seems that none of the existing platforms use a secure protocol to establish if their subchain is consistent. This is a serious issue, because it either limits the expressiveness of the smart contracts supported by these platforms (which must consider all messages as consistent, so basically losing the notion of state), or degrades the security of contracts (because adversaries can manage to publish inconsistent messages, so tampering with the execution of smart contracts).

Contributions The protocol presented here allows meta-nodes to maintain a consistent subchain over the Bitcoin blockchain. The protocol is based on *Proof-of-Stake* [16, 44], since extending the subchain must be endorsed with a money deposit. Intuitively, a meta-node which publishes a consistent message gets back its deposit once the message is confirmed by the rest of the network. In particular, the protocol provides an economic incentive to honest meta-nodes, while disincentivizing the dishonest ones.

Notably, this protocol can be implemented in Bitcoin by only using the so-

called *standard* transactions[1].

7.2 Background: Bitcoin and the blockchain

Bitcoin is a cryptocurrency and a digital open-source payment infrastructure that has recently reached a market capitalization of over $90 billions[2]. The Bitcoin network is peer-to-peer, not controlled by any central authority [65]. Each Bitcoin user owns one or more personal wallets, which consist of pairs of asymmetric cryptographic keys: the public key uniquely identifies the user *address*, while the private key is used to authorize payments. *Transactions* describe transfers of bitcoins (Ƀ), and the history of all transactions, which recorded on a public, immutable and decentralised data structure called *blockchain*, determines how many bitcoins are contained in each address.

To explain how Bitcoin works, consider two transactions $\mathtt{t}_0$ and $\mathtt{t}_1$, which can be graphically represented as follows:[3]

$\mathtt{t}_0$
in: $\cdots$ in-script: $\cdots$
out-script$(\mathtt{t}, \sigma)$: $ver_k(\mathtt{t}, \sigma)$ value: v_0

$\mathtt{t}_1$
in: $\mathtt{t}_0$ in-script: $sig_k(\bullet)$
out-script$(\cdots)$: $\cdots$ value: v_1

The transaction $\mathtt{t}_0$ contains v_0Ƀ, which can be *redeemed* by putting on the blockchain a transaction (e.g., $\mathtt{t}_1$), whose in field is the cryptographic hash of the whole $\mathtt{t}_0$ (for simplicity, just displayed as $\mathtt{t}_0$ in the figure). To redeem $\mathtt{t}_0$, the in-script of $\mathtt{t}_1$ must contain values making the out-script of $\mathtt{t}_0$ (a boolean programmable function) evaluate to true. When this happens, the value of $\mathtt{t}_0$ is transferred to the new transaction $\mathtt{t}_1$, and $\mathtt{t}_0$ is no longer redeemable. Similarly, a new transaction can then redeem $\mathtt{t}_1$ by satisfying its out-script.

In the example displayed above, the out-script of $\mathtt{t}_0$ evaluates to true when receiving a digital signature σ on the redeeming transaction t, with a given key pair k. Let $ver_k(\mathtt{t}, \sigma)$ denote the signature verification, and $sig_k(\bullet)$ the signature of the enclosing transaction ($\mathtt{t}_1$ in the example above), including *all* the parts of the transaction *except* its in-script.

Now, assume that the blockchain contains $\mathtt{t}_0$, not yet redeemed, when someone tries to append $\mathtt{t}_1$. To validate this operation, the nodes of the Bitcoin network check that $v_1 \leq v_0$, and then they evaluate the out-script of $\mathtt{t}_0$, by instantiating its formal parameters t and σ, to $\mathtt{t}_1$ and to the signature $sig_k(\bullet)$, respectively. The function ver_k verifies that the signature is correct: therefore, the out-script succeeds, and $\mathtt{t}_1$ redeems $\mathtt{t}_0$.

Bitcoin transactions may be more general than the ones illustrated by the previous example: their general form is displayed in Figure 7.1. First, there can be multiple inputs and outputs (denoted with array notation in the figure).

[1] This is important, because non-standard transactions are discarded by nodes running the official Bitcoin client.

[2] Source: crypto-currency market capitalizations http://coinmarketcap.com, as of October 2017

[3] in-script and out-script are respectively referred as scriptPubKey and scriptSig in the Bitcoin documentation.

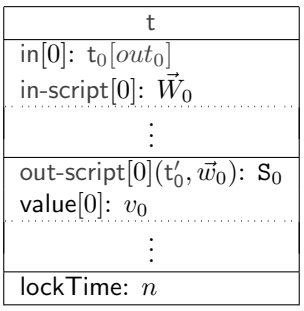


Figure 7.1: General form of transactions.

Each output has an associated out-script and value, and can be redeemed independently from others. Consequently, in fields must specify which output they are redeeming ($\mathrm{t}_0[out_0]$ in the figure). Similarly, a transaction with multiple inputs associates an in-script to each of them. To be valid, the sum of the values of all the inputs must be greater or equal to the sum of the values of all outputs. In its general form, the out-script is a program in a (not Turing-complete) scripting language, featuring a limited set of logic, arithmetic, and cryptographic operators. Finally, the lockTime field specifies the earliest moment in time (block number or UNIX timestamp) when the transaction can appear on the blockchain.

The Bitcoin network is populated by a large set nodes, called *miners*, which collect transactions from clients, and are in charge of appending the valid ones to the blockchain. To this purpose, each miner keeps a local copy of the blockchain, and a set of unconfirmed transactions received by clients, which it groups into *blocks*. The goal of miners is to add these blocks to the blockchain, in order to get a revenue. Appending a new block B_i to the blockchain requires miners to solve a cryptographic puzzle, which involves the hash $h(B_{i-1})$ of block B_{i-1}, a sequence of unconfirmed transactions $\langle T_i \rangle_i$, and some salt R. More precisely, miners have to find a value of R such $h(h(B_{i-1}) \| \langle T_i \rangle_i \| R) < \mu$, where the value μ is adjusted dynamically, depending on the current hashing power of the network, to ensure that the average mining rate is of 1 block every 10 minutes. The goal of miners is to win the "lottery" for publishing the next block, i.e. to solve the cryptopuzzle before the others; when this happens, the miner receives a reward in newly generated bitcoins, and a small fee for each transaction included in the mined block. If a miner claims the solution of the current cryptopuzzle, the others discard their attempts, update their local copies of the blockchain with the new block B_i, and start mining a new block on top of B_i. In addition, miners are asked to verify the validity of the transactions in B_i by executing the associated scripts. Although verifying transactions is not mandatory, miners are incentivized to do that, because if in any moment a transaction is found invalid, they lose the fee earned if the transaction was published in the blockchain.

If two or more miners solve a cryptopuzzle simultaneously, they create a *fork* in the blockchain (i.e., two or more parallel valid branches). In the presence of a fork, miners must choose a branch wherein carrying out the mining process; roughly, this divergence is resolved once one of the branches becomes longer than the others. When this happens, the other branches are discarded, and all the orphan transactions contained therein are nullified.

Overall, this protocol essentially implements a *"Proof-of-Work"* system [32].

7.3 Background: Subchains and consistency

Let $\{\mathsf{A}, \mathsf{B}, \ldots\}$ be a set of participants, who want to append messages $\mathtt{a}, \mathtt{b}, \ldots$ to the subchain. A *label* is a pair containing a participant A and a message $\mathtt{a}$, written $\mathsf{A} : \mathtt{a}$. *Subchains* are finite sequences of labels, written $\mathsf{A}_1 : \mathtt{a}_1 \cdots \mathsf{A}_n : \mathtt{a}_n$, which are embedded in the Bitcoin blockchain. The intuition is that A_1 has embedded the message $\mathtt{a}_1$ in some transaction $\mathtt{t}_1$ of the Bitcoin blockchain, then A_2 has appended some transaction $\mathtt{t}_2$ embedding $\mathtt{a}_2$, and so on. For a subchain η, let $\eta\ \mathsf{A} : \mathtt{a}$ be the subchain obtained by appending $\mathsf{A} : \mathtt{a}$ to η.

In general, labels can also have side effects on the Bitcoin blockchain: let $\mathsf{A} : \mathtt{a}(v \to \mathsf{B})$ represent a label which also transfers v฿ from A to B. When this message is on the subchain, it also acts as a standard currency transfer on the Bitcoin blockchain, which makes v฿ in a transaction of A redeemable by B. When the value v is zero or immaterial, the notation is simplified to $\mathtt{a}$ instead of $\mathtt{a}(v \to \mathsf{B})$.

A crucial insight is that not all possible sequences of labels are valid subchains: to define the *consistent* ones, subchains are interpreted as traces of *Labelled Transition Systems* (LTS). Formally, an LTS is a tuple $(Q, L, q_0, \to)$, where:

- Q is a set of states (ranged over by $q, q', \ldots$);
- L is a set of labels (in our case, of the form $\mathsf{A} : \mathtt{a}$);
- $q_0 \in Q$ is the initial state;
- $\to \subseteq Q \times L \times Q$ is a transition relation.

As usual, write $q \xrightarrow{\mathsf{A}:\mathtt{a}} q'$ when $(q, \mathsf{A} : \mathtt{a}, q') \in \to$, and, given a subchain $\eta = \mathsf{A}_1 : \mathtt{a}_1 \cdots \mathsf{A}_n : \mathtt{a}_n$, the notation $q \xrightarrow{\eta} q'$ is used whenever there exist $q_1, \ldots, q_n$ such that:

$$q \xrightarrow{\mathsf{A}_1:\mathtt{a}_1} q_1 \xrightarrow{\mathsf{A}_2:\mathtt{a}_2} \cdots \xrightarrow{\mathsf{A}_n:\mathtt{a}_n} q_n = q'$$

The relation $\to$ is required to be *deterministic*, i.e. if $q \xrightarrow{\mathsf{A}:\mathtt{a}} q'$ and $q \xrightarrow{\mathsf{A}:\mathtt{a}} q''$, then it must be $q' = q''$.

The intuition is that the subchain has a state (initially, q_0), and each message updates the state according to the transition relation. More precisely, if the subchain is in state q, then a message $\mathtt{a}$ sent by A makes the state evolve to q' whenever $q \xrightarrow{\mathsf{A}:\mathtt{a}} q'$ is a transition in the LTS.

Note that, for some state q and label $\mathsf{A} : \mathtt{a}$, it may happen that no state q' exists such that $q \xrightarrow{\mathsf{A}:\mathtt{a}} q'$. In this case, if q is the current state of the subchain, the goal is to make hard for a participant (possibly, an adversary trying to tamper with the subchain) to append such message. Informally, a subchain $\mathsf{A}_1 : \mathtt{a}_1 \cdots \mathsf{A}_n : \mathtt{a}_n$ is *consistent* if, starting from the initial state q_0, it is possible to find states $q_1, \ldots, q_n$ such that from each q_i there is a transition labelled $\mathsf{A}_{i+1} : \mathtt{a}_{i+1}$ to q_{i+1}.

Definition 7.1 (Subchain consistency)**.** A subchain η is said *consistent* whenever there exists q such that $q_0 \xrightarrow{\eta} q$.

Note that, if a subchain is consistent, then by determinism it follows that the state q_n exists and is unique. In other words, a consistent sequence of messages uniquely identifies the state of the subchain.

Example 7.3.1. To illustrate consistency, consider a smart contract $\texttt{FACTORS}_n$ which rewards with 1Đ each participant who extends the subchain with a new prime factor of n. The contract accepts two kinds of messages:

- $\texttt{send}_p$, where p is a natural number;
- $\texttt{pay}_p(1 \to \mathsf{A})$, meaning that A receives a reward for the factor p;

The states of the contract can be represented as sets of triples (A, p, b), where b is a boolean value indicating whether A has been rewarded for the factor p. The initial state is $\emptyset$. The transition relation of $\texttt{FACTORS}_n$ is defined as follows:

- $S \xrightarrow{\mathsf{A}:\texttt{send}_p} S'$, iff p is a prime factor of n, $(\mathsf{B}, p, b) \notin S$ for any B and b, and $S' = S \cup \{(\mathsf{A}, p, 0)\}$;
- $S \xrightarrow{\mathsf{F}:\texttt{pay}_p(1\to\mathsf{A})} S'$, iff $(\mathsf{A}, p, 0) \in S$ and $S' = (S \setminus \{(\mathsf{A}, p, 0)\}) \cup \{(\mathsf{A}, p, 1)\}$.

Consider now the following subchains for $\texttt{FACTORS}_{330}$, where F is the participant who issues the contract, and M is an adversary:

1. $\eta_1 = \mathsf{A}:\texttt{send}_{11} \quad \mathsf{B}:\texttt{send}_2 \quad \mathsf{F}:\texttt{pay}_{11}(1 \to \mathsf{A}) \quad \mathsf{F}:\texttt{pay}_2(1 \to \mathsf{B})$
2. $\eta_2 = \mathsf{A}:\texttt{send}_{11} \quad \mathsf{F}:\texttt{pay}_{11}(1 \to \mathsf{A}) \quad \mathsf{M}:\texttt{send}_{11}$
3. $\eta_3 = \mathsf{M}:\texttt{send}_{229} \quad \mathsf{F}:\texttt{pay}_{229}(1 \to \mathsf{M})$
4. $\eta_4 = \mathsf{A}:\texttt{send}_{11} \quad \mathsf{F}:\texttt{pay}_{11}(1 \to \mathsf{M})$

The subchain η_1 is consistent, because both A and B send new factors and get their rewards. The subchains η_2 and η_3 are inconsistent, because 11 sent by M is not fresh, and 229 is not a factor of 330. Finally, the subchain η_4 is inconsistent, because M gets the reward that should have gone to A. □

Similarly to Bitcoin, the aim is not guaranteeing that a subchain is *always* consistent. Indeed, also in Bitcoin a miner could manage to append a block with invalid transactions: in this case, as discussed in Section 7.2, the Bitcoin blockchain forks, and the other miners must choose which branch to follow. However, honest miners will neglect the branch with invalid transactions, so eventually (since honest miners detain the majority of computational power), that branch will be abandoned by all miners.

For subchain consistency notion adopted is similar: it is assumed that an adversary can append a label $\mathsf{A}:\mathtt{a}$ such that $q_n \not\xrightarrow{\mathsf{A}:\mathtt{a}}$, so making the subchain inconsistent. However, upon receiving such label, honest nodes will discard it. To formalise their behaviour, a function Γ is defined below that, given a subchain η (possibly inconsistent), filters all the invalid messages. Hence, $\Gamma(\eta)$ is a consistent subchain.

Definition 7.2 (Branch pruning)**.** The endofunction Γ is inductively defined on subchains as follows, where ϵ denotes the empty subchain:

$$\Gamma(\epsilon) = \epsilon \qquad \Gamma(\eta\ \mathsf{A}:\mathtt{a}) = \begin{cases} \Gamma(\eta)\ \mathsf{A}:\mathtt{a} & \text{if } \exists q, q' : q_0 \xrightarrow{\Gamma(\eta)} q \xrightarrow{\mathsf{A}:\mathtt{a}} q' \\ \Gamma(\eta) & \text{otherwise} \end{cases}$$

In order to model which labels can be appended to the subchain without breaking its consistency, the auxiliary relation $\models$ is introduced below. Informally, given a consistent subchain η, the relation $\eta \models \mathsf{A}:\mathtt{a}$ holds whenever the subchain $\eta\,\mathsf{A}:\mathtt{a}$ is still consistent.

Definition 7.3 (Consistent update)**.** $\mathsf{A}:\mathtt{a}$ is a *consistent update* of a subchain η, denoted with $\eta \models \mathsf{A}:\mathtt{a}$, iff the subchain $\Gamma(\eta)\ \mathsf{A} : \mathtt{a}$ is consistent.

Example 7.3.2. Recall the subchain $\eta_2 = \mathsf{A}:\mathtt{send}_{11}\ \mathsf{F}:\mathtt{pay}_{11}(1 \to \mathsf{A})\ \mathsf{M}:\mathtt{send}_{11}$ from Example 7.3.1. Then $\mathsf{B}:\mathtt{send}_2$ is a consistent update of η_2, because $\Gamma(\eta_2)\ \mathsf{B}:\mathtt{send}_2 = \mathsf{A}:\mathtt{send}_{11}\ \mathsf{F}:\mathtt{pay}_{11}(1 \to \mathsf{A})\ \mathsf{B}:\mathtt{send}_2$ is consistent. □

7.4 A protocol for consensus on Bitcoin subchains

Assume a network of mutually distrusted nodes $\mathsf{N}, \mathsf{N}', \ldots$, called *meta-nodes* to distinguish them from the nodes of the Bitcoin network. Meta-nodes receive messages from participants (also mutually distrusting) which want to extend the subchain. The goal is to allow honest participants (i.e., those who follow the protocol) to append consistent updates to the subchain, while disincentivizing adversaries who attempt to make the subchain inconsistent.

To this purpose, this protocol is based on *Proof-of-Stake* (PoS) with the assumption that the overall stake retained by honest participants is greater than the stake of dishonest ones[4]. The stake is needed by meta-nodes, which have to vote to approve messages sent by participants. These messages are embedded into Bitcoin transactions, and called *update requests*. Let $\mathsf{UR}[\mathsf{A}:\mathtt{a}]$ denote the update request issued by A to append the message $\mathtt{a}$ to the subchain. In order to vote an update request, a meta-node must invest ν₿ on it, where ν is a constant specified by the protocol. An update request requires the vote of a single meta-node. The protocol requires meta-nodes to vote a request $\mathsf{UR}[\mathsf{A}:\mathtt{a}]$ only if $\mathsf{A}:\mathtt{a}$ is a consistent update of the current subchain η, i.e. if $\eta \models \mathsf{A}:\mathtt{a}$[5]. To incentivize meta-nodes to vote their update requests, participants pay them a constant *fee*, which can be redeemed by meta-nodes when the update request is appended to the subchain.

The protocol is defined in Figure 7.2, and is organised in *stages*. The protocol ensures that *exactly one* label $\mathsf{A}:\mathtt{a}$ is appended to the subchain for each stage i. This is implemented by appending a corresponding transaction $\mathsf{UR}_i[\mathsf{A}:\mathtt{a}]$ to the Bitcoin blockchain. To guarantee its uniqueness, the protocol exploits an *arbiter* T, namely a distinguished node of the network which is assumed honest (this hypothesis is discussed in Section 7.5.1). The main steps of the protocol are now described.

At step 1 of the stage i of the protocol, a meta-node (say, N) votes an update request (as detailed in Section 7.5.2).

[4]Note that a similar hypothesis, but related to computational power rather than stake, holds in Bitcoin, where honest miners are supposed to control more computational power than dishonest ones.

[5]It is assumed that all meta-nodes agree on the Bitcoin blockchain; since η is a projection of the blockchain, they also agree on η.

1. Upon receiving an update request UR[A : a], a meta-node checks its consistency, $\eta \models A : a$. If so, it votes the request, and adds it to the request pool;
2. when Δ expires, the arbiter signs all the well-formed URs in the request pool;
3. all requests signed by the arbiter are sent to the Bitcoin miners, to be published on the blockchain. The first to be mined, indicated with UR_i, becomes the i-th label of the subchain.

Figure 7.2: Summary of a protocol stage i.

In order to do this, N must confirm some of the past C updates (where $C \geq 1$ is the *cutoff window*, a constant fixed by the protocol and described in Section 7.4.1). To confirm an update, N uses the ν₿ to pay the meta-nodes who respectively appended each chosen update UR_j (with $i - C \leq j < i$) to the subchain. The way to choose the updates UR_j to be confirmed is called *refund policy* and is deepened in Section 7.4.1. After voting, N adds UR[A : a] to the *request pool*, i.e. the set of all voted requests of the current stage (emptied at the beginning of each stage). This voting step has a fixed duration Δ, specified by the protocol (the choice of Δ is discussed in Section 7.6).

At step 2, which starts when Δ expires, the arbiter T signs all *well-formed* request transactions, i.e., those respecting the format defined in Section 7.5.2.

At step 3, meta-nodes send the requests signed by T to the Bitcoin network. The mechanism described in Section 7.5.2 ensures that, at each stage i, exactly one transaction, denoted UR_i[A : a], is put on the Bitcoin blockchain. When this happens, the label A : a is appended to the subchain.

Summarizing, the protocol depends on the parameters $\Pi = (C, \nu, f, r)$, which are, respectively, the cutoff window size, the amount required to vote an update request, the fee payed by the client, and the maximum transferable amount in special updates in the form $A : a(v \to B)$ (where, by definition, $v \leq r$).

7.4.1 Refund policies

A refund policy can be formally defined as a function Θ that, given a subchain η and the protocol parameters $\Pi = (C, \nu, f, r)$, outputs a sequence of refunds $\rho^i = (\rho^i_1 \dots \rho^i_C)$, where:

- ρ^i_j represents, at each stage i of the protocol, the amount to pay to the meta-nodes who voted UR_{i-j}, for every j s.t. $1 \leq j \leq C$ (only updates inside the current cutoff window can be refunded);
- $\sum_{j=1}^{C} \rho^i_j = \nu + f$, that is the policy specifies how to split the vote and the fee among the voters of the updates inside the cutoff window.

To enforce good behaviour, updates whose voters did not follow the prescribed policy are considered not *refundable*. This means honest meta-nodes penalize not only inconsistent labels, but also illegal refunds.

Definition 7.4 (Refundable update)**.** Let $\eta^{[1\dots k]}$ be the subchain after the completion of the k-th protocol stage, let $\mathsf{UR}_j[\mathsf{A} : \mathsf{a}]$ be a published update, and $\bar{\rho}^j$ the refund made by its voter. Then, UR_j is said *refundable* if and only if it is consistent ($\eta^{[1\dots(j-1)]} \models \mathsf{A} : \mathsf{a}$) and it follows the refund policy ($\bar{\rho}^j = \Theta(\eta^{[1\dots(j-1)]}, \Pi)$).

Thus, at each stage i, it is possible to define the set of indexes of refundable updates in the current cutoff window. Suppose that the subchain starts with C predetermined and consistent updates $\mathsf{UR}_{-i}, 1 \leq i \leq C$, then:

$$\xi^0 = \{1 \leq j \leq C\} \tag{7.1}$$

since all initial updates are considered refundable. Then, for $i > 0$:

$$\xi^i = \{1 \leq j \leq C : \mathsf{UR}_{i-j} \text{ is refundable according to } \Theta\} \tag{7.2}$$

Note that checking if the voter of the j-th update (with $j < i$) has followed the refund policy just requires to examine the updates with index $k \leq j$. Thus, this check may depend only on ξ^k and never on ξ^i that therefore is well-defined.

As an example, some possible refund policies are now presented.

newest-first This policy refunds only the newest refundable update in the cutoff window (if any), the newest in general otherwise:

$$\rho^i_j = \begin{cases} \nu + f & \text{if } \xi^i \neq \emptyset \wedge j = min(\xi^i) \\ \nu + f & \text{if } \xi^i = \emptyset \wedge j = 1 \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

oldest-first This policy refunds the oldest consistent update (if any), the oldest in general otherwise (note that it coincides with the `newest-first` if $C = 1$):

$$\rho^i_j = \begin{cases} \nu + f & \text{if } \xi^i \neq \emptyset \wedge j = max(\xi^i) \\ \nu + f & \text{if } \xi^i = \emptyset \wedge j = C \\ 0 & \text{otherwise} \end{cases} \tag{7.4}$$

Proof-of-Burn

To expand the possibilities for meta-nodes and increase the security of the protocol, as will be deepened in Section 7.5.1, the sequence of refunds ρ can be extended to include a special value ρ_0. This value represents the amount that should be paid to a pre-set fictional address (e.g. an *all-zero* address). Refunding such an address effectively corresponds to burning the money sent, making it unspendable.

With this enhancement, the policies previously defined can be improved, removing the case $\xi^i = \emptyset$ and adding:

$$\rho^i_0 = \begin{cases} \nu + f & \text{if } \xi^i = \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{7.5}$$

which can be interpreted as follows: if no update in the cutoff window is refundable, burn vote and fee. The variants of the previous policies, after

the inclusion of the Proof-of-Burn condition, are called `newest-first-pburn` and `oldest-first-pburn`. This change avoids the (forced) confirmation of a non-refundable update, that is allowed in the previous definitions of the `newest-first` and `oldest-first` policies.

Now, recall that in Section 7.4 the condition $C \geq 1$ is provided. However, the choice $C = 1$ makes sense only if there is the possibility of burning the vote. Vice versa, voters would have no other choice besides confirming the previous update (refundable or not). Introducing the Proof-of-Burn, instead, the following policy for $C = 1$ can be defined and used.

harsh policy This policy refunds the previous update if refundable, burns the money otherwise:

$$\rho_1^i = \begin{cases} \nu + f & \text{if } \xi^i = \{1\} \\ 0 & \text{otherwise} \end{cases} \qquad \rho_0^i = \begin{cases} \nu + f & \text{if } \xi^i = \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{7.6}$$

7.5 Evaluation of the protocol

In this section the basic properties of the protocol will be shown and its security evaluated, providing some experimental results. In particular, a real-world attack scenario will be illustrated, and it will be investigated how the choice of protocol parameters and refund policy can affect the protocol security.

It will also considered how possible attacks to Bitcoin may affect subchains built on top of its blockchain.

7.5.1 Basic properties of the protocol

Now some basic properties of the protocol will be established. Hereafter, it will be assumed that honest nodes control the majority of the total stake of the network, denoted by S. Further, assume that the overall stake required to vote pending update requests is greater than the overall stake of honest meta-nodes.

Adversary power An honest meta-node votes as many requests as is allowed by its stake. Hence, if its stake is h, it votes h/ν requests per stage. Consequently, the rest of the network (which may include dishonest meta-nodes not following the protocol) can vote at most $(S - h)/\nu$ requests[6]. Given these assumptions this proposition follows:

Proposition 7.5.1. The probability that an honest meta-node with stake h updates the subchain is at least h/S at each stage.

Since it is assumed that honest meta-nodes control the majority of the stake, Proposition 7.5.1 also limits the capabilities of the adversary:

Proposition 7.5.2. If the global stake of honest meta-nodes is S_H, then dishonest ones update the subchain with probability at most $(S - S_H)/S$ at each stage.

[6]Note that assuming the ability of the adversary to delay some messages, thus reducing the honest meta-nodes effective voting power (since their voted requests might not reach the request pool), is equivalent to consider an adversary with a higher stake.

Although inconsistent updates are ignored by honest meta-nodes, their side effects as standard Bitcoin transactions (i.e. trasfers of vɃ from A to B in labels $\mathsf{A} : \mathtt{a}(v \to \mathsf{B})$) cannot be revoked once they are included in the Bitcoin blockchain.

Even though the goal of the protocol is to let meta-nodes get revenues proportionate to their probability of updating the subchain (as defined in Proposition 7.5.1 and Proposition 7.5.2), the adversary might exploit these side effects to earn more than due by publishing inconsistent updates. Therefore, it will be shown how the incentive system in the protocol reduces the feasibility of such inconsistent updates.

According to Proposition 7.5.2, if M has stake m, and the other meta-nodes are honest, then M has probability at most m/S of extending the subchain in a given stage of the protocol. Since stages can be seen as independent events, it follows that:

Proposition 7.5.3. The probability that an adversary with stake m saturates a cutoff window with her updates only (consistent or not) is μ^C, where C is the cutoff window size, and $\mu = m/S$.

To simplify the terminology, hereafter a consistent update will be considered to also be refundable[7].

Now, assume that M manages to publish C consecutive updates (consistent or inconsistent) starting from index j, with probability given by Proposition 7.5.3. M can use each update at index $j < k \leq j + C$ to recover the vote ν and collect the fee f for the previous update at index $k - 1$, such that only the last update at index $j + C$ remains unrefunded.

In particular, if the protocol specifies a refund policy that does not admit the *Proof-of-Burn* described in Section 7.4.1, at least a honest update at index $i > j+C$ has to necessarily refund M of $(\nu+f)$Ƀ, since the cutoff window is saturated with updates published by M only. Consequently, following this strategy the attacker does not lose any deposit and possibly earns an additional extra revenue r for each inconsistent update published, if any. This extra revenue r models the case where M induces a victim A to publish an inconsistent update in the form $\mathsf{A} : \mathtt{a}(r \to \mathsf{M})$.

Also note that, if M cannot manage to saturate the cutoff window immediately, the completion of the attack can be delayed by publishing at least one inconsistent update every C ones on the subchain (to keep refunding vote and fee to themself). The above behaviour of M (and all its variants) is called *self-compensation attack*.

Finally, observe that the choice of the protocol parameters and, particularly, the refund policy is crucial to force the honest strategy to be more profitable than any dishonest one. To support this claim, in what follows it is shown a dishonest strategy which exploits a variant of the attack, called the *reversed self-compensation attack*, and it will be proven that it is always more profitable than the honest strategy when the chosen refund policy is `newest-first`.

Reversed self-compensation attack Assume an adversary M that manages to append two updates on the subchain, the first with index i, and the second

[7] Publishing a consistent update that is not refundable does not break the consistency of the subchain, but it causes the meta-node who voted the update to be (eventually) not refunded for its effort. Therefore, this behaviour cannot be considered an attack.

with index $i+1 < j \leq i+C$. Suppose that the update at index i is consistent, then the honest meta-node that publishes the update at index $i+1$ refunds $(\nu + f)$Ɓ to M (recall, the considered refund policy is `newest-first`). Now M can use again these funds to publish a new inconsistent update at index j, refunding again the update at index i (thus also violating the refund policy). So M manages to earn the undeserved extra revenue r without losing neither ν nor f, therefore performing a special case of the self-compensation attack.

Now, consider a conservative strategy where the attacker M at first tries to publish consistent updates only. When such an update is published, M switches to trying to publish just one inconsistent update, and tries to do so until success or the last consistent update published is beyond the cutoff window, then reverses again to publishing consistent updates.

Let μ be the probability M has in successfully publishing an update, and suppose that the last update is a consistent update published by M. It will be shown that the expected payoff ϕ_D that M earns following the described dishonest strategy, is *always* greater than the expected payoff ϕ_H that would be earned following the honest strategy. The analysis is limited to the subsequent C updates, since the two strategies coincide afterwards, and for $C \geq 2$ (with $C = 1$, meta-nodes have no choice besides refunding the last update, so an inconsistent update is always more profitable than a consistent one). From the hypothesis, it follows that:

$$\phi_D = \mu f + \sum_{i=1}^{C-1} \mu(1-\mu)^{i-1}(f + r + \mu f(C-1-i)) \tag{7.7}$$

$$\phi_H = \mu f C \tag{7.8}$$

Then, the gain M obtains by following the dishonest strategy is:

$$\begin{aligned}
\phi_D - \phi_H &= \mu f(1-C) + \mu \sum_{i=1}^{C-1} (1-\mu)^{i-1}(f + r + \mu f(C-1-i)) \\
&= \mu f \left(\sum_{i=1}^{C-1} (1-\mu)^{i-1}(1 + \mu(C-1-i)) - (C-1) \right) + \mu r \sum_{i=1}^{C-1} (1-\mu)^{i-1} \\
&= \mu r \sum_{i=1}^{C-1} (1-\mu)^{i-1}
\end{aligned} \tag{7.9}$$

The result of eq. (7.9) is justified by the following Claim 7.1. Since $0 < \mu < 1$, $\forall r$ s.t. $r > 0$ it follows that $\phi_D > \phi_H$. This means that, indipendently from the chosen protocol parameters Π, a protocol that uses the refund policy `newest-first` admits at least one dishonest strategy which is always more profitable than the honest one. Note also that a similar result can be obtained for the `oldest-first` policy.

Claim 7.1. *For $C \geq 2$, it holds:*

$$\sum_{i=1}^{C-1} (1-\mu)^{i-1}(1 + \mu(C-1-i)) - (C-1) = 0 \tag{7.10}$$

Proof.

$$
\begin{aligned}
&\sum_{i=1}^{C-1}(1-\mu)^{i-1}(1+\mu(C-1-i))-(C-1)\\
&=\sum_{i=0}^{C-2}(1-\mu)^{i}(1+\mu(C-2-i))-(C-1)\\
&=\sum_{i=0}^{C-2}\left(\sum_{j=0}^{i}\binom{i}{j}(-\mu)^j\right)(1+\mu(C-2-i))-(C-1)\\
&=\sum_{j=0}^{C-2}(-\mu)^j\left(\sum_{i=j}^{C-2}\binom{i}{j}(1+\mu(C-2-i))\right)-(C-1)\\
&=\sum_{j=0}^{C-2}\left((-\mu)^j\sum_{i=j}^{C-2}\binom{i}{j}-(-\mu)^{j+1}\sum_{i=j}^{C-2}\binom{i}{j}(C-2-i)\right)-(C-1)\\
&=(-\mu)^0\sum_{i=0}^{C-2}\binom{i}{0}-(C-1)+\sum_{j=1}^{C-2}(-\mu)^j\sum_{i=j}^{C-2}\binom{i}{j}-\sum_{j=1}^{C-1}(-\mu)^j\sum_{i=j-1}^{C-2}\binom{i}{j-1}(C-2-i)\\
&=\sum_{j=1}^{C-2}(-\mu)^j\left(\sum_{i=j}^{C-2}\binom{i}{j}-\sum_{i=j-1}^{C-3}\binom{i}{j-1}(C-2-i)\right)\\
&=\sum_{j=1}^{C-2}(-\mu)^j\left(\sum_{i=j}^{C-2}\binom{i}{j}-\sum_{i=j}^{C-2}\binom{i-1}{j-1}(C-1-i)\right)\\
&=\sum_{j=1}^{C-2}(-\mu)^j\sum_{i=j}^{C-2}\left(\binom{i}{j}-\binom{i-1}{j-1}(C-1-i)\right) \qquad (7.11)
\end{aligned}
$$

From the following Claim 7.2, it follows that the coefficients of the polynomial in Equation (7.11) are all zero, thus proving the above Claim 7.1. □

Claim 7.2. *For $n \geq 1$, it holds:*

$$\sum_{i=j}^{n}\left(\binom{i}{j}-\binom{i-1}{j-1}(n+1-i)\right)=0 \qquad 1\leq j\leq n \qquad (7.12)$$

Proof. We prove it by induction over n. The base case is $n=1$, therefore $j=1$.

$$\sum_{i=1}^{1}\left(\binom{i}{1}-\binom{i-1}{0}(2-i)\right)=1-1=0 \qquad (7.13)$$

For the inductive step:

$$\begin{aligned}
&\sum_{i=j}^{n+1}\left(\binom{i}{j}-\binom{i-1}{j-1}(n+2-i)\right)\\
&=\sum_{i=j}^{n+1}\left(\binom{i}{j}-\binom{i-1}{j-1}(n+1-i)\right)-\sum_{i=j}^{n+1}\binom{i-1}{j-1}\\
&=\sum_{i=j}^{n}\left(\binom{i}{j}-\binom{i-1}{j-1}(n+1-i)\right)+\binom{n+1}{j}-\sum_{i=j}^{n+1}\binom{i-1}{j-1}\\
&=\binom{n+1}{j}-\sum_{i=j}^{n+1}\binom{i-1}{j-1}
\end{aligned}$$

To conclude, the results of the following Lemma 7.1 are needed. □

Lemma 7.1. *For $n \geq 1$ it holds:*

$$\binom{n}{k}=\sum_{i=k}^{n}\binom{i-1}{k-1} \qquad 1 \leq k \leq n \tag{7.14}$$

Proof. The proof is again by induction over n.The base case is $n = 1$, thus $k = 1$.

$$\binom{1}{1}=1=\sum_{i=1}^{1}\binom{0}{0} \tag{7.15}$$

For the inductive step:

$$\begin{aligned}
\binom{n+1}{k}&=\binom{n}{k}+\binom{n}{k-1} \qquad 1 \leq k \leq n \\
&=\binom{n}{k-1}+\sum_{i=k}^{n}\binom{i-1}{k-1}\\
&=\sum_{i=k}^{n+1}\binom{i-1}{k-1}
\end{aligned} \tag{7.16}$$

□

Trustworthiness of the arbiter The protocol uses an arbiter T to ensures that exactly one transaction per stage is appended to the blockchain, and that its choice is random as well. In order to simplify the description of the protocol, the arbiter T has been assumed to behave honestly. However, the arbiter does not play the role of a trusted authority: indeed, the update requests to be voted are chosen by the meta-nodes, and once they are added to the request pool, the arbiter is expected to sign all of them, without taking part on the validation nor in the voting. Since everyone can inspect the request pool, any misbehaviour of the arbiter can be detected by the meta-nodes, that can proceed to replace it.

Adversary strategy To analyse possible attacks, consider an adversary who can craft any update (consistent or not), and controls one meta-node M with stake ratio $\mu = m/S$, where $\mu \in [0; 1]$, m is the stake controlled by the adversary and S is the total stake of the network[8]. Suppose that each meta-node can vote as many update requests as possible, spending all its stake, and that the network is always saturated with pending updates, which globally amount to the entire stake of honest meta-nodes[9].

To evaluate its security, the protocol is modelled as a game, in which the attacker M is a player that adopts a possibly dishonest strategy, thus trying to publish either consistent or inconsistent updates. Conversely, the other players are the honest meta-nodes, that follow the protocol and therefore adopt a honest strategy, trying to publish consistent updates only. Suppose also that M follows an *optimal* strategy, i.e. at each protocol stage the choice whether to vote a consistent or inconsistent update is made with the goal of maximizing the revenue, given the current state of the subchain. Note that the current state mainly depends on the content of the current cutoff window:

Proposition 7.5.4. The revenue of an update published by an adversary M, at the protocol stage i, depends only on the state of the cutoff window in that stage (i.e., $\eta^{[(i-C)\ldots(i-1)]}$), on the protocol parameters Π, on the refund policy Θ and on the adversary ratio μ.

To justify Proposition 7.5.4, observe that, by definition of refund policy Θ, no update with index $j < i - C$ can be refunded, neither of the vote ν nor the fee f, in a protocol stage with index i. Thus, no matter what M chooses at the current stage, there is no additional revenue (but also no loss) for any update outside the cutoff window.

Now, let G be a function that maps a subchain $\eta^{[1\ldots k]}$ into a sequence of labels $s = \sigma_1 :: \ldots :: \sigma_k$. A label σ_i can assume one of the following values: `Inc`, which indicates an inconsistent update published by M; `Con`, which represents a consistent update published by M, and `Ext`, that denotes a (consistent) external update published by the rest of the network, assumed to be honest.

Also, let $s_C^k = G(\eta^{[(k-C)\ldots(k-1)]})$ be the sequence that represents the cutoff window state at the stage k This sequence is used to generate two new sequences $s^k_{\texttt{Con}} = s_C^k :: \texttt{Con}$ and $s^k_{\texttt{Inc}} = s_C^k :: \texttt{Inc}$ that represent the possible continuations of the chain if the adversary manages to publish the next update.

Now, let ϕ be a function that, given the protocol parameters Π and the refund policy Θ, takes a sequence s as input and computes the *a posteriori* attacker revenue associated to s. Moreover, let $\bar{\phi}$ be a variation of the same function that also takes in consideration the possible refunds given by C subsequent `Ext` updates (that necessarily follow the prescribed refund policy). This variation models what happens when the attack terminates.

Through ϕ and $\bar{\phi}$, and given the attacker ratio μ, it is possible to define a payoff function Φ_d with depth d. Let $s = \sigma_1 :: \ldots :: \sigma_N$ be a sequence of length

[8] Assuming a single adversary is not less general than having many non-colluding meta-nodes which carry on individual attacks. Indeed, in this setting meta-nodes do not join their funds to increase the stake ratio μ.

[9] Note that saying the update queue is not always saturated is equivalent to model an adversary with a stronger μ: this because honest meta-nodes cannot spend all their stake in a single protocol stage, i.e. reducing their actual *power*. Thus, studying this particular case will not give any additional contribution to the analysis.

N, and $s' = \sigma_2 :: \ldots :: \sigma_N$ be the same sequence of s, where the first label is elided. The payoff function is defined recursively as:

$$\Phi_d(s) = \begin{cases} \phi(s) + (1-\mu)\Phi_{d-1}(s' :: \texttt{Ext}) + \mu \max(\Phi_{d-1}(s' :: \texttt{Con}), \Phi_{d-1}(s' :: \texttt{Inc})) & d > 1 \\ \bar{\phi}(s) & d = 1 \end{cases}$$

Remember that, for the Proposition 7.5.4, the revenue of the current update depends on the state of the cutoff window (and not on the full history of the subchain).

Definition 7.5 (Optimal choice)**.** Given a sequence ss_C^k that represents the state of the current cutoff window, the *optimal choice with depth d* for an attacker is to try to publish a consistent update if $\Phi_d(s^k_{\texttt{Con}}) > \Phi_d(s^k_{\texttt{Inc}})$, an inconsistent one otherwise.

Consider an adversary that plans to meddle with the protocol for a limited amount of time, say for d stages starting at the stage n. Then the optimal strategy would be at each stage $n \leq k \leq n + d - 1$ to take the optimal choice with depth $n + d - k$. In fact the optimal choice with depth d (i.e. the first choice) considers every possible evolution of the protocol for the duration of the attack, weighted for its probability to occur, and considers the best outcome at every step, thus maximizes the revenue. Now consider an attacker that plans to attack for an indefinite amount of stages, then the optimal strategy is not well defined, but can be approximated by taking at each step the optimal choice with fixed depth d, if d is high enough.

Analytic results In what follows, the results of the security analysis of the protocol are shown, under the attack scenario in which the adversary adopts an optimal strategy, and only for the `harsh-policy`, which provides the best security performance among the policies shown in Section 7.4.1 (according to some simulated preliminary experiments shown in Figure 7.4). The results are summarized by the following:

Proposition 7.5.5. If the protocol uses the `harsh-policy`, an adversary with stake ratio μ that adopts an optimal strategy in pursuing an attack of arbitrary length, remains necessarily honest if and only if $\mu < 1 - r/(\nu + f)$.

Proof. Note that, when the prescribed policy is the `harsh-policy`, any update after `Con` necessarily refunds the vote and the fee to the adversary, independently from its type. So this gain can be included in the revenue, having:

- $\phi(-, \texttt{Con}) = f$

where the symbol '$-$' indicates an indifference condition.

On the other side, the revenue for an inconsistent update can be quantified in:

- $\phi(\texttt{Ext}, \texttt{Inc}) = r - \nu$
- $\phi(\texttt{Inc}, \texttt{Inc}) = r + f$ (vote and fee are self-refunded)
- $\phi(\texttt{Con}, \texttt{Inc}) = r - \nu$ (the refund of f and ν has already been counted for `Con`)

Note that, when considering whether to publish an inconsistent update, a cutoff that contains `Con` is equivalent to one with `Ext`. Finally, observe that computing the revenue in this way gives $\phi(-, \texttt{Ext}) = 0$, and therefore $\phi = \bar{\phi}$.

Given these considerations, it is easy to see that in the terminal stages of the attack the adversary is always encouraged to publish `Con`, since from the condition $\nu > r$ it follows that at least two consecutive `Inc` are required to outdo the honest behaviour. Only exceptions are the rare cases in which a streak of consecutive `Inc` reaches the end of the attack: in this situation the adversary is obviously motivated to continue trying to publish `Inc`. But as soon as an attempt to publish fails and an `Ext` insinuates itself in the sequence, the adversary will become honest if the end is near enough.

On the contrary, at the early stages a high enough probability μ could ensure that, on average, a sufficient number of consecutive `Inc` will be published to gain an advantage over a honest behaviour. But, if this is the case, note that $\phi(\texttt{Inc}, \texttt{Inc}) \geq \phi(-, \texttt{Inc})$ so if `Inc` is the optimal choice at the step k, then it is the optimal choice at the step $k - 1$ too.

So any optimal strategy is either always honest, or starts completely dishonest (always tries to publish `Inc`) and becomes honest near the end. This means that for the purposes of this analysis it is sufficient to consider the early stages, when an `Inc` may still be more profitable than a `Con`. In this period the adversary publishing an `Inc` will gain $\phi(\texttt{Ext}, \texttt{Inc})$ with probability $1 - \mu$ and $\phi(\texttt{Inc}, \texttt{Inc})$ with probability μ, since μ is the probability that the adversary actually manages to publish an update.

So the adversary remains honest right from the start if and only if:

$$\begin{aligned}
&\phi(-, \texttt{Con}) > (1 - \mu) \cdot \phi(\texttt{Ext}, \texttt{Inc}) + \mu \cdot \phi(\texttt{Inc}, \texttt{Inc}) \\
&\iff f > r - \nu + (f + \nu)\mu \\
&\iff f > r - \nu + (f + \nu)\mu \\
&\iff \mu < \frac{f + \nu - r}{f + \nu} \\
&\iff \mu < 1 - \frac{r}{f + \nu} \qquad (7.17)
\end{aligned}$$

□

The value of f is assumed to be strictly smaller than ν, in order to incentivize the participation of meta-nodes to the protocol. In fact, with f very close to (or even greater than) ν, clients have no evident benefit from delegating meta-nodes to vote their updates (since the economical effort does not change significantly). This is similar to provide a protocol with no fees, which is less attractive for meta-nodes to participate in. However, a large participation to the protocol reduces the possibility that single or colluding entities controls the majority of the stake.

Under this assumption, the eq. (7.17) shows that the security of the protocol is essentially proportional to the ratio ν/r: the higher this ratio, the more convenient an honest behaviour becomes. Figure 7.3 finally illustrates how the *switch point* (i.e., the $min(\mu)$ such that $\phi_{\texttt{Inc}} \geq \phi_{\texttt{Con}}$) varies for different combinations of ν and r, with a small fixed fee ($f = 0.01$Ð).

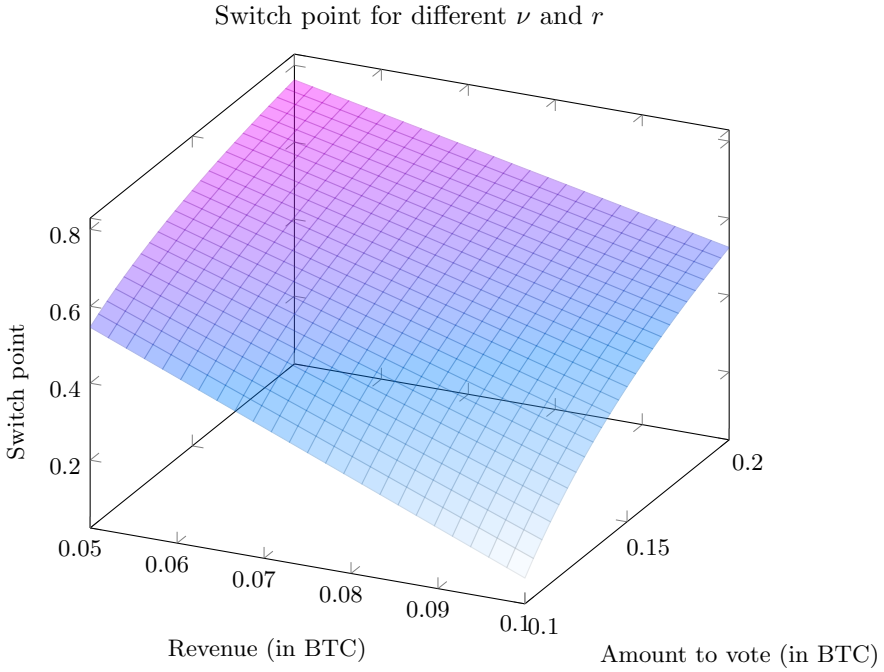


Figure 7.3: The plot shows how the switch point (the $min(\mu)$ such that $\phi_{\texttt{Inc}} \geq \phi_{\texttt{Con}}$) varies for different combinations of ν and r, when the refund policy is `harsh-policy` and f is fixed to 0.01Ƀ.

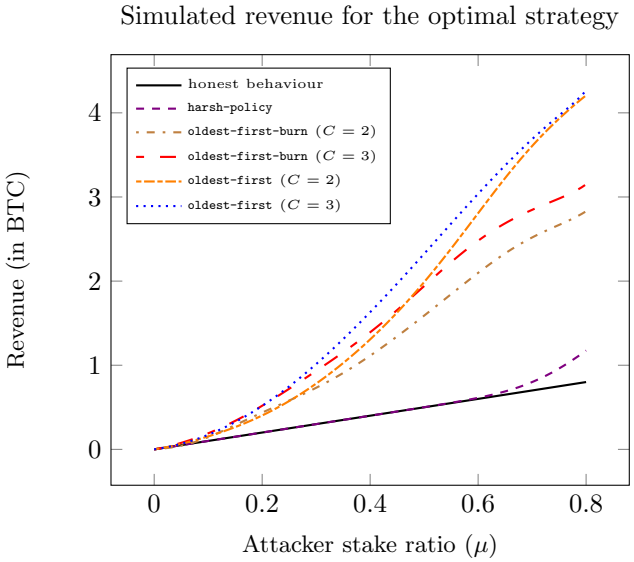


Figure 7.4: We simulated the revenue of an adversary M participates in subchains of length 100 and uses the optimal strategy. Each curve represents the revenue when μ increases, and for a different refund policy. Protocol parameters are fixed ($f = 0.01$Ƀ, $r = 0.03$Ƀ and $v = 0.1$Ƀ), and the results are the mean taken over 10000 iterations.

Security of the underlying Bitcoin blockchain So far only direct attacks to the protocol have been considered, assuming the underlying Bitcoin blockchain to be secure. However, although Bitcoin has been secure in practice till now, some works have spotted some potential vulnerabilities of its protocol. These vulnerabilities could be exploited to execute *Sybil attacks* [8] and *selfish-mining attacks* [36], which might also affect subchains built on top of the Bitcoin blockchain.

In Sybil attacks on Bitcoin, honest nodes are induced to believe that the network is populated by many distinct participants, which instead are controlled by a single malicious entity. This attack is usually exploited to quickly propagate malicious information on the network, and to disguise honest participants in a consensus/reputation protocol, e.g. by overwhelming the network with votes of the adversary. In the selfish-mining attack [36], small groups of colluding miners manage to obtain a revenue larger than the one of honest miners. More specifically, when a selfish-mining pool finds a new block, it keeps it hidden to the rest of the network. In this way, selfish miners gain an advantage over honest ones in mining the next block. This is equivalent to keep a private fork of the blockchain, which is only known to the selfish-mining pool. Note that honest miners still mine on the public branch of the blockchain, and their hash rate is greater than selfish miners' one. Since, in the presence of a fork, the Bitcoin protocol requires to keep mining on the longest chain, selfish miners reveal their private fork to the network just before being overcome by the honest miners. Eyal and Sirer in [36] show that, under certain assumptions, this strategy gives better revenues than honest mining: in the worst scenario (for the adversary), the attack succeeds if the selfish-mining pool controls at least 1/3 of the total hashing power. Rational miners are thus incentivized to join the selfish-mining pool. Once the pool manages to control the majority of the hashing power, the system loses its decentralized nature. Garay, Kiayias and Leonardos in [37] essentially confirm these results: considering a core Bitcoin protocol, they prove that if the hashing power γ of honest miners exceeds the hashing power β of the adversary pool by a factor λ, then the ratio of adversary blocks in the blockchain is bounded by $1/\lambda$ (which is strictly greater than β). Thus, as β (the adversary pool size) approaches 1/2, they control the blockchain.

Although these attacks are mainly related to Bitcoin revenues, they can affect the consistency of any subchain built on top of its blockchain. In particular, suitably adapted versions of these attacks allow adversaries to cheat meta-nodes about the current subchain state, forcing them to synchronize their local copy of the Bitcoin blockchain with invalid forks that will be discarded by the network in the future. To protect against such attacks, meta-nodes should consider only *l-confirmed* transactions. Namely, if the last published blockchain block is B_n, they consider only those transactions appearing in blocks B_j with $j \leq n - l$. This means that an attacker would have to mine at least l blocks to force the revocation of a *l-confirmed* transaction. Rosenfeld [73] shows that, if an attacker controls at most the 10% of the network hashing power, $l = 6$ is sufficient for reducing the risk of revoking a transaction to less than 0.1%.

7.5.2 Implementation in Bitcoin

In this section it is shown how the protocol can be implemented in Bitcoin. A label $\mathsf{A} : \mathtt{a}(v \to \mathsf{B})$ at position i of the subchain is implemented as the Bitcoin

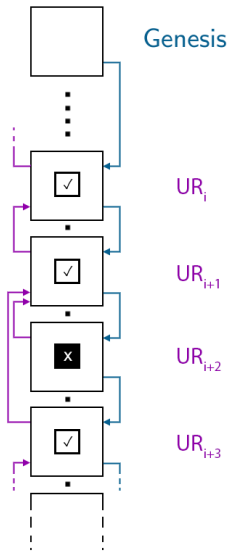
transaction $\mathsf{UR}_i[\mathsf{A}:\mathtt{a}(v \to \mathsf{B})]$ in Figure 7.5a, with the following outputs:

- the output of index 0 embeds the label $\mathsf{A}:\mathtt{a}$. This is implemented through an unspendable `OP_RETURN` script [11][10].
- the output of index 1 links the transaction to the previous element of the subchain, pointed by in[2]. This link requires the arbiter signature. Note that, since all the update requests in the same stage redeem the same output, exactly one of them can be mined.
- the output of index 2 implements the incentive mechanism. The script rewards the meta-node N' which has voted a preceding UR_j in the subchain. Meta-node N' can redeem from this output κɃ plus the participant's fee, by providing his signature.
- the output of index 3 is only relevant for messages $\mathtt{a}(v \to \mathsf{B})$ where $v > 0$. Participant B can redeem vɃ from this output by providing his signature.

Genesis
in: $\cdots$
in-script: $\cdots$
out-script(t, σ): $ver_{\mathsf{T}}(\mathsf{t}, \sigma)$
value: 0.0001

$\mathsf{UR}_i[\mathsf{A}:\mathtt{a}(v \to \mathsf{B})]$
in[0]: Fee_i
in-script[0]: $sig_{\mathsf{C}}(\bullet)$
in[1]: Stake_i
in-script[0]: $sig_{\mathsf{N}}(\bullet)$
in[2]: $\mathsf{Confirm}_{i-1}$ (at index 1)
in-script[0]: $sig_{\mathsf{T}}(\bullet)$
out-script[0](): `OP_RETURN` $\mathsf{A}:\mathtt{a}$
value[0]: 0
out-script[1](t, σ): $ver_{\mathsf{T}}(\mathsf{t}, \sigma)$
value[1]: 0.0001
out-script[2](t, σ): $ver_{\mathsf{N}'}(\mathsf{t}, \sigma)$
value[2]: ν + fee
out-script[3](t, σ): $ver_{\mathsf{B}}(\mathsf{t}, \sigma)$
value[3]: v
lockTime: $n + k$

(a)



(b)

Figure 7.5: In (a), format of Bitcoin transactions used to implement the protocol. In (b), a subchain mantained through our protocol. Since UR_{i+2} contains an inconsistent update, the meta-node which voted it is not rewarded.

All transactions specify a lockTime $n + k$, where n is the current Bitcoin block number, and k is a positive constant. This ensures that a transaction can be mined only after k blocks. In this way, even if a transaction is signed by the

[10] The `OP_RETURN` instruction allows to save 80 bytes metadata in a transaction; an out-script containing `OP_RETURN` always evaluates to false, hence it is unspendable.

arbiter and sent to miners before the others, it has the same probability as the others of being appended to the blockchain.

To initialise the subchain, the arbiter puts the Genesis transaction on the Bitcoin blockchain. This transaction secures a small fraction of bitcoin, which can be redeemed by UR_1 through the arbiter signature. This value is then transferred to each subsequent update of the subchain (see Figure 7.5b). At each protocol stage, participants send incomplete UR transactions to the network. These transactions contain only in[0] and out[0], specifying the fee and the message for the subchain (including the value to be transferred). To vote, meta-nodes add in[1], in[2] and out[2] to these transactions, to, respectively, put the required ν (from some transaction Stake_i), declare they want extend the last published update $\mathsf{Confirm}_{i-1}$, and specify the previous update to be rewarded. All the in[1] fields in a stage of the protocol must be different, to prevent attackers to vote more URs with the same funds.

7.6 Discussion

The protocol presented aims to reach consensus on subchains, i.e. chains of platform-dependent messages embedded in the Bitcoin blockchain. The protocol incentivizes nodes to validate messages before appending them to the subchain, making economically disadvantageous for an adversary to append inconsistent messages. To confirm this intuition the security of the protocol has been measured over different attack scenarios, showing that, under conservative assumptions, its security is comparable to that of Bitcoin.

Performance of the protocol. As seen in Section 7.4, the protocol runs in periods of duration Δ. Due to the mechanism for choosing the message to append to the subchain from the request pool, the protocol can publish at most one transaction per Bitcoin block. This means that a lower bound for Δ is the Bitcoin block interval ($\sim$10mins). To monitor the arbiter behaviour throughout protocol stages, all meta-nodes must share a coherent view of the request pool. Then, Δ needs to be large enough to let each node synchronize the request pool with the rest of the network. A possible approach to cope with this issue is to make meta-nodes broadcast their voted updates, and to keep a list of other ones (considering only those which satisfy the format of transactions, as in Section 7.5.2). More efficient approaches could exploit distributed shared memories [25, 42].

Overcoming the metadata size limit. As noted in Section 7.5.2, OP_RETURN unspendable scripts are used to embed metadata in Bitcoin transactions. Since Bitcoin limits the size of such metadata to 80 bytes, this might not be enough to store the data needed by platforms. To overcome this issue, one can use distributed hash tables [61] maintained by meta-nodes. In this way, instead of storing full message data in the blockchain, OP_RETURN scripts would contain only the corresponding message digests. The unique identifier of the Bitcoin transaction can be used as the key to retrieve the full message data from the hash table.

Smart contracts over subchains. The model of subchains defined in Section 7.3, based on LTSs, can be easily extended to model the computations of smart contracts over the Bitcoin blockchains. A platform for smart contracts could exploit this model to represent the state of a contract as the state of the subchain, and model its possible state updates through the transition relation.

Implementing a platform for smart contracts would require a language for expressing them. To bridge this language with the abstract model presented, one can provide the language with an operational semantics, giving rise to an LTS describing the computations. Note that the assumption to model computations as a single LTS does not reduce the generality of the system, since a set of LTSs, each one modelling a contract, can be encoded in one LTS as their parallel composition. If the language is Turing-complete, an additional problem is the potential non-termination. This issue has been dealt with in different ways by different platforms. E.g., the approach followed by Ethereum [35] is to impose a fee for each instruction executed by its virtual machine. If the fee does not cover the cost of the whole computation, the execution terminates.

A usable platform must also allow to create new contracts at run-time. Since in this model the LTS representing possible computations is fixed, it is necessary to introduce a mechanism to "extend" it. To handle the publication of new contracts, the protocol could be modified so that UR may contain its code, and the unique identifier of the transaction also identifies the contract. In this extended model, update requests would also contain the identifier of the contract to be updated, so that meta-nodes can execute the corresponding code.

Chapter 8

Conclusions

The applications of cryptography in the digital world are very variegated, and new protocols are proposed almost every day. In this jungle of possibilities it is difficult to navigate and choose solutions that actually satisfy the application's requirements.

To help distinguishing between hot air and decent algorithms, proofs of security are great allies. In fact a proof gives a minimum warranty on the effectiveness of the protocol, with results based on hard logic instead of subjective sensations.

The protocols presented in this thesis address different use cases and illustrate the diversity of applications and solutions. The proofs given show their strength against powerful attackers in realistic scenarios, starting from well-established assumptions on hard problems, either on famous algebraic problems or widely used cryptographic primitives. Thanks to these qualities these schemes become good candidates as real-life solutions.

A trait that can be found in all these protocols, although very different one from the other, is the fragility of the equilibriums between assumptions, attack scenario and the protocol. Very slight variations that may seem innocuous may destroy a proof. An apparently equivalent solution in the description of a protocol could force the use of a way stronger assumption or a much weaker adversary, or even make the proof impossible. As the old saying goes, *the devil is in the details.*

Bibliography

[1] GMP library. `https://gmplib.org/`.

[2] Making sense of blockchain smart contracts. `http://www.coindesk.com/making-sense-smart-contracts/`. Last accessed 2017/01/14.

[3] OP_RETURN statistics. `http://opreturn.org/`. Last accessed 2016/12/15.

[4] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. Fair two-party computations via Bitcoin deposits. In *Financial Cryptography Workshops*, pages 105–121, 2014.

[5] R. Aragona, R. Longo, and M. Sala. Several proofs of security for a tokenization algorithm. *Applicable Algebra in Engineering, Communication and Computing*, pages 1–12, 2017.

[6] M. Araoz and E. Ordano. Proof of existence. *Online. Available: proofofexistence. com*, 2013.

[7] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. De Panafieu, C. Ràfols, et al. Attribute-based encryption schemes with constant-size ciphertexts. *Theoretical Computer Science*, 422:15–38, 2012.

[8] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar. On Bitcoin and red balloons. In *ACM Conference on Electronic Commerce (EC)*, pages 56–73, 2012.

[9] W. Banasik, S. Dziembowski, and D. Malinowski. Efficient zero-knowledge contingent payments in cryptocurrencies without scripts. In *ESORICS*, volume 9879 of *LNCS*, pages 261–280. Springer, 2016.

[10] M. Bartoletti, S. Lande, and A. S. Podda. A proof-of-stake protocol for consensus on Bitcoin subchains. In B. et al, editor, *Financial Cryptography and Data Security*, pages 568–584, Cham, 2017. Springer International Publishing.

[11] M. Bartoletti and L. Pompianu. An analysis of Bitcoin OP_RETURN metadata. In *Financial Cryptography Workshops*, 2017. Also available as CoRR abs/1702.01024.

[12] M. Bartoletti and R. Zunino. Constant-deposit multiparty lotteries on Bitcoin. In *Financial Cryptography Workshops*, 2017. Also available as IACR Cryptology ePrint Archive 955/2016.

[13] A. Beimel. *Secure schemes for secret sharing and key distribution.* PhD thesis, Technion-Israel Institute of technology, Faculty of computer science, 1996.

[14] M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-Preserving Encryption. *Selected Areas in Cryptography – SAC 2009*, LNCS 5867:295–312, 2009.

[15] M. Bellare, P. Rogaway, and T. Spies. The FFX mode of operation for Format-Preserving Encryption (Draft 1.1). *Manuscript (standards proposal) submitted to NIST*, 2010.

[16] I. Bentov, A. Gabizon, and A. Mizrahi. Cryptocurrencies without proof of work. In *Financial Cryptography Workshops*, volume 9604 of *LNCS*, pages 142–157. Springer, 2016.

[17] I. Bentov and R. Kumaresan. How to use Bitcoin to design fair protocols. In *CRYPTO*, volume 8617 of *LNCS*, pages 421–439. Springer, 2014.

[18] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Proc. of SP 07*, pages 321–334, 2007.

[19] BitID. Bitid open protocol. Technical report, 2015. `http://bitid.bitcoin.blue/`, Last Accessed 2017/03/14.

[20] J. Black and P. Rogaway. Ciphers with Arbitrary Finite Domains. volume LNCS 2271, pages 114–130. Springer, 2002.

[21] Blockstore: Key-value store for name registration and data storage on the Bitcoin blockchain. `https://github.com/blockstack/blockstore`, 2014.

[22] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Proc. of EUROCRYPT 05*, volume 3494 of *LNCS*, pages 440–456. 2005.

[23] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology—CRYPTO 2001*, pages 213–229. Springer, 2001.

[24] E. Brier, T. Peyrin, and J. Stern. BPS: a Format-Preserving Encryption proposal. *Manuscript (standards proposal) submitted to NIST*, 2010.

[25] M. Cai, A. Chervenak, and M. Frank. A peer-to-peer replica location service based on a distributed hash table. In *ACM/IEEE Conference on High Performance Networking and Computing*, page 56. IEEE Computer Society, 2004.

[26] M. Chase. Multi-authority attribute based encryption. In *Theory of Cryptography*, pages 515–534. Springer, 2007.

[27] M. Chase and S. S. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 121–130. ACM, 2009.

[28] D. Cooper and et al. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. Technical report, IETF RFC 5280, 2008.

[29] K. Crary and M. J. Sullivan. Peer-to-peer affine commitment using Bitcoin. In *ACM PLDI*, pages 479–488, 2015.

[30] R. Dermody, A. Krellenstein, O. Slama, and E. Wagner. Counter-Party: Protocol specification. `http://counterparty.io/docs/protocol_specification/`, 2014.

[31] S. Díaz-Santiago, L. M. Rodríguez-Henríquez, and D. Chakraborty. A Cryptographic Study of Tokenization Systems. *International Journal of Information Security*, 15(4):413–432, 2016.

[32] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *CRYPTO*, volume 740 of *LNCS*, pages 139–147. Springer, 1993.

[33] T. Elgamal. A public key cryptosystem and a signature scheme based on discret e logaritmhs. *IEEE Trans. on Inf. Th.*, 31(4):469–472, 1985.

[34] EMVCo. Payment Tokenisation Specification - Technical Framework, Version 1.0. Technical report, March 2014.

[35] Ethereum Foundation. Ethereum project. Technical report, 2015. `https://www.ethereum.org/`, Last accessed 2017/03/14.

[36] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, volume 8437 of *LNCS*, pages 436–454. Springer, 2014.

[37] J. A. Garay, A. Kiayias, and N. Leonardos. The Bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT*, volume 9057 of *LNCS*, pages 281–310. Springer, 2015.

[38] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. of CCS 06*, pages 89–98, 2006.

[39] A. Hern. A history of Bitcoin hacks. *The Guardian*, march 2014. `http://www.theguardian.com/technology/2014/mar/18/history-of-bitcoin-hacks-alternative-currency` Last accessed 2017/03/14.

[40] V. T. Hoang, B. Morris, and P. Rogaway. An Enciphering Scheme Based on a Card Shuffle. *Advances in Cryptology – CRYPTO 2012*, LNCS 7417:1–13, 2012.

[41] S. Hohenberger and B. Waters. Attribute-based encryption with fast decryption. In *Proc. of PKC 13*, volume 7778 of *LNCS*, pages 162–179. 2013.

[42] S. Iyer, A. Rowstron, and P. Druschel. Squirrel: A decentralized peer-to-peer web cache. In *PODC*, pages 213–222. ACM, 2002.

[43] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ECDSA). Technical report, Certicom, 1998.

[44] A. Kiayias, I. Konstantinou, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure Proof-of-Stake blockchain protocol. *IACR Cryptology ePrint Archive*, 2016:889, 2016.

[45] A. Kiayias, H. Zhou, and V. Zikas. Fair and robust multi-party computation using a global transaction ledger. In *EUROCRYPT*, pages 705–734, 2016.

[46] R. Kumaresan and I. Bentov. How to use Bitcoin to incentivize correct computations. In *ACM CCS*, pages 30–41, 2014.

[47] R. Kumaresan, T. Moran, and I. Bentov. How to use Bitcoin to play decentralized poker. In *ACM CCS*, pages 195–206, 2015.

[48] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Proc. of EUROCRYPT 10*, volume 7881 of *LNCS*, pages 62–91. 2010.

[49] A. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *Theory of Cryptography*, volume 5978 of *LNCS*, pages 455–479. 2010.

[50] A. Lewko and B. Waters. Decentralizing attribute-based encryption. In *Proc. of EUROCRYPT 11*, volume 6632 of *LNCS*, pages 568–588. 2011.

[51] X. Liang, Z. Cao, H. Lin, and J. Shao. Attribute based proxy re-encryption with delegating capabilities. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, pages 276–286, New York, NY, USA, 2009. ACM.

[52] H. Lin, Z. Cao, X. Liang, and J. Shao. Secure threshold multi authority attribute based encryption without a central authority. *Information Sciences*, 180(13):2618–2632, 2010.

[53] M. Liskov, R. L. Rivest, and D. Wagner. Tweakable Block Ciphers. *Journal of Cryptology*, 24(3):588–613, 2011.

[54] Z. Liu and Z. Cao. On efficiently transferring the linear secret-sharing scheme matrix in ciphertext-policy attribute-based encryption. *IACR Cryptology ePrint Archive*, 2010.

[55] R. Longo, C. Marcolla, and M. Sala. Key-policy multi-authority attribute-based encryption. In *Maletti A. (eds) Algebraic Informatics. CAI 2015. Lecture Notes in Computer Science, vol 9270*, pages 152–164. Springer International Publishing, 2015.

[56] R. Longo, C. Marcolla, and M. Sala. Collaborative multi-authority key-policy attribute-based encryption for shorter keys and parameters. Cryptology ePrint Archive, Report 2016/262, 2016. `https://eprint.iacr.org/2016/262`,accepted and presented at CAI 2017 International Conference.

[57] R. Longo, F. Pintore, G. Rinaldo, and M. Sala. On the security of the blockchain BIX protocol and certificates. In *9th International Conference on Cyber Conflict: Defending the Core*, CyCon, pages 217–232. NATO CCD COE Publications, 2017.

[58] B. Lynn. PBC library. `https://crypto.stanford.edu/pbc/`.

[59] B. Lynn. *On the implementation of pairing-based cryptosystems*. PhD thesis, Stanford University, 2007.

[60] S. Matsumoto and R. M. Reischuk. Ikp: Turning a pki around with blockchains. Cryptology ePrint Archive, Report 2016/1018, 2016. `https://eprint.iacr.org/2016/1018`.

[61] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *Workshop on Peer-to-Peer Systems (IPTPS)*, volume 2429 of *LNCS*, pages 53–65. Springer, 2002.

[62] B. Morris, P. Rogaway, and T. Stegers. How to Encipher Messages on a Small Domain. *Advances in Cryptology – CRYPTO 2009*, LNCS 5677:286–302, 2009.

[63] S. Muftic. Bix certificates: Cryptographic tokens for anonymous transactions based on certificates public ledger. *Ledger*, 1:19–37, 2016.

[64] S. Müller, S. Katzenbeisser, and C. Eckert. Distributed attribute-based encryption. In *Information Security and Cryptology–ICISC 2008*, pages 20–36. Springer, 2009.

[65] S. Nakamoto. Bitcoin: a peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.

[66] NIST. Secure Hash Standard (SHS). *FIPS PUB 180-4*, 2015.

[67] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *Proc. of CCS 07*, pages 195–203, 2007.

[68] P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. *LNCS*, 3788:11–20, 2005.

[69] S. PCI. Information Supplement: PCI DSS Tokenization Guidelines, Version 2.0. Technical report, August 2011.

[70] B. Preneel. The state of cryptographic hash functions. *LNCS*, 1561:158–182, 1999.

[71] M. O. Rabin. Digital signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT laboratory for computer science, 01 1979.

[72] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[73] M. Rosenfeld. Analysis of hashrate-based double spending. *CoRR*, abs/1402.2009, 2014.

[74] T. Ruffing, A. Kate, and D. Schröder. Liar, liar, coins on fire!: Penalizing equivocation by loss of Bitcoins. In *ACM CCS*, pages 219–230, 2015.

[75] A. Rukhin and et al. A Statistical Test Suite for the Validation of Random and Pseudo Random Number Generators for Cryptographic Applications. *NIST Special Publication*, 2010.

[76] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473. Springer, 2005.

[77] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in cryptology*, pages 47–53. Springer, 1985.

[78] P. SSC. Tokenization Product Security Guidelines - Irreversible and Reversible Tokens, Version 1.0. Technical report, April 2015.

[79] P. SSC. PCI DSS Requirements and Security Assessment Procedures, Version 3.2. Technical report, April 2016.

[80] E. Stefanov and E. Shi. Fastprp: Fast pseudo-random permutations for small domains. Cryptology ePrint Archive, Report 2012/254, 2012. `https://eprint.iacr.org/2012/254`.

[81] N. Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997. `http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/548`.

[82] A. Tomescu and S. Devadas. Catena: Efficient non-equivocation via Bitcoin. In *IEEE Symp. on Security and Privacy*, 2017.

[83] M. Vanhoef and F. Piessens. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *Proceedings of the ACM Conference on Computer and Communications Security, Dallas, TX, USA*, volume 30, 2017.

[84] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *Proc. of CRYPTO 09*, volume 5677 of *LNCS*, pages 619–636. 2009.

[85] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Proc. of PKC 11*, volume 6571 of *LNCS*, pages 53–70. 2011.

[86] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin. Storj: A peer-to-peer cloud storage network. Technical report, 2014. `http://storj.io/storj.pdf` Last accessed 2017/03/14.

[87] XMPP Standards Foundation. Extensible messaging and presence protocol. Technical report, 2015. `https://www.xmpp.org/` Last accessed 2017/03/14.